



INTERWOVEN

# **TeamSite® Templating Developer's Guide**

**Release 5.5.1**

© 1999-2002 Interwoven, Inc. All rights reserved.

No part of this publication (hardcopy or electronic form) may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Interwoven. Information in this manual is furnished under license by Interwoven, Inc. and may only be used in accordance with the terms of the license agreement. If this software or documentation directs you to copy materials, you must first have permission from the copyright owner of the materials to avoid violating the law which could result in damages or other remedies.

Interwoven, TeamSite, OpenDeploy and the Interwoven logo are trademarks of Interwoven, Inc., which may be registered in certain jurisdictions. SmartContext, DataDeploy, Content Express, OpenChannel, OpenSyndicate, MetaTagger, TeamCatalog, TeamXML, TeamXpress, the tagline and service mark are trademarks of Interwoven, Inc. which may be registered in certain jurisdictions. All other trademarks are owned by their respective owners.

TeamSite Templating utilizes third party components under the following copyrights with all rights reserved: Copyright 1999, Apache Software Foundation ([www.apache.org](http://www.apache.org)); Copyright 1997-2000, Samizdat Productions. If you are interested in using these components for other purposes, contact the appropriate vendor.



**INTERWOVEN**

Interwoven, Inc.

803 Eleventh Ave.

Sunnyvale, CA 94089

<http://www.interwoven.com>

Printed in the United States of America

Version 5.5.1

Part # 40-00-10-12-00-551-200

# Table of Contents

## About This Book 7

- Manual Organization 7
- Notation Conventions 9
- Editing Text on Windows NT Systems 10
- Notation of iw-home on UNIX and Windows NT Systems 10

## Chapter 1: TeamSite Templating Installation 11

- Hardware Requirements 11
- Operating System Requirements 11
- Pre-Installation Validation 12
- Installing on Solaris 12
- Installing on Windows NT 14
- Installing on Client Machines 15
  - Memory Requirements for TeamSite Templating Java Clients 15
  - Setting Heap Size 16
  - Multiple Users on a Single Java TeamSite Templating Client 16
- Next Step 16

## Chapter 2: Initial Configuration 17

- Configuration Overview 17
  - Concepts and Definitions 18
  - Process Flow: Creating a New Data Content Record 26
  - Process Flow: Generating an Output File 28
  - The Example Directory Structure 30
- Configuring the Example Templating Environment 31
  - Editing available\_templates.cfg to Initiate Workflows 33
  - Modifying the TeamSite iw.cfg File 34
- Providing Content Contributor Interface Access 35
- Starting TeamSite Templating 36

## **Chapter 3: Setting Up Data Capture Templates 39**

User Interface Overview	39
Data Capture Template Overview	40
Example Data Capture Templates	42
Data Capture Example 1	42
Example 1 Data Capture Form	42
Example 1 datacapture.cfg File	45
Explanation of datacapture.cfg File	47
Customizing the Appearance of Java Data Capture Forms	66
Example 1 Data Content Record	68
Data Capture Example 2	70
Example 2 Data Capture Form	70
Example 2 datacapture.cfg File	72
Data Capture Template DTD	76

## **Chapter 4: Setting Up Presentation Templates 83**

Creating Presentation Templates	83
Using a Presentation Template—An Example	87
Interwoven XML Tags	96

## **Chapter 5: Mapping Users, Templates, and Content Records 97**

templating.cfg Overview	97
Example templating.cfg File	98
Diagram Key	101
Setting Previewing Path Variables	103
templating.cfg DTD	105

## **Chapter 6: Integrating Templating, DataDeploy, and Workflow 107**

Integration Overview	108
Integration Steps	108
TeamSite Templating	108
DataDeploy	109
TeamSite Workflow	109

## **Appendix A: Using Callouts 111**

- The Java Callout 111
  - The datacapture.cfg File 112
  - Java Source Code 112
- The CGI Callout 116
  - The Data Capture Form 116
  - The datacapture.cfg.example File 118
  - The example\_datacapture\_callout.ipl File 119

## **Appendix B: Command-Line Tools 129**

- iwddctacleval 130
- iwdtd2sym 132
- iwgen 133
- iwprop 134
- iwpt\_compile.ipl 135
- iwregen 138
- iwxml\_validate.ipl 139
- upgrade\_dct\_cfg.ipl 141

## **Appendix C: Creating DCTs from DTDs 145**

- Running the CLT on the DTD File 146
- The datacapture.cfg File 147
  - Diagram Key 149
- Unsupported DTD Features 150

## **Appendix D: Internationalization 153**

- Limitations 154
- Japanese EUC-JP Encoding Support 155
- Localized Java Templating into Japanese, Traditional Chinese, and Simplified Chinese 157

## **Index 159**



# About This Book

---

The *TeamSite Templating Developer's Guide* is a guide to installing and configuring TeamSite® Templating and to developing presentation templates and data capture templates. It is primarily intended for TeamSite Templating developers and for web server administrators and system administrators. Many of the operations described in this manual require root (UNIX®) or Administrator (Windows NT®) access to the TeamSite server. If you do not have root or Administrator access to the TeamSite server, consult your system administrator.

UNIX: Users should be familiar with basic UNIX commands and be able to use an editor such as emacs or vi.

Windows NT: Users should be familiar with either IIS or Netscape® web servers, and with basic Windows NT operations such as adding users and modifying ACLs (Access Control Lists).

It is also helpful to be familiar with regular expression syntax. If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions*, by Jeffrey Friedl.

For information about TeamSite, refer to the *TeamSite User's Guide* and the *TeamSite Administration Guide*.

## Manual Organization

This manual contains the following sections and chapters:

- Chapter 1, “TeamSite Templating Installation,” describes how to install TeamSite Templating on Solaris and Windows NT systems.
- Chapter 2, “Initial Configuration,” describes how to configure the fully functional example templating environment supplied with TeamSite Templating.
- Chapter 3, “Setting Up Data Capture Templates,” describes how to customize data capture templates for your site-specific needs.



- Chapter 4, “Setting Up Presentation Templates,” describes how to customize presentation templates for your site-specific needs.
- Chapter 5, “Mapping Users, Templates, and Content Records,” describes how to configure user access to specific templates and data content records, which templates can interact with which data content records, and other configurable features of TeamSite Templating.
- Chapter 6, “Integrating Templating, DataDeploy, and Workflow,” describes how to configure TeamSite Templating to work with DataDeploy and TeamSite workflow.
- Appendix A, “Using Callouts,” describes the cgi-callout and java-callout elements.
- Appendix B, “Command-Line Tools,” describes how users can use TeamSite Templating through command-line commands.
- Appendix C, “Creating DCTs from DTDs,” describes how to convert existing industry-standard DTDs into data capture templates for use in TeamSite Templating.



## Notation Conventions

This manual uses the following notation conventions:

Convention	Definition and Usage
<b>Bold</b>	Text that appears in a GUI element (for example, menu items, buttons, or elements of a dialog box) and command names are shown in bold. For example:  Click <b>Edit File</b> in the Button Bar.
<i>Italics</i>	Book titles appear in italics. Terms are italicized the first time they are introduced. Valuable information may be italicized for emphasis.
Monospace	Commands, command-line output, and file names are in monospace type. For example:  The <code>iwextattr</code> command-line tool allows you to set and look up extended attributes on a file.
<i>Monospaced italic</i>	Monospaced italics are used for command-line variables. For example: <code>iwckrole <i>role user</i></code> This means that you must replace <i>role</i> and <i>user</i> with your values.
<b>Monospaced bold</b>	Monospaced bold represents information you enter in response to system prompts. The character that appears before a line of user input represents the command prompt, and should not be typed. For example:  <code><b>iwextattr -s project=proj1 //IWSERVER/default/main/dev/WORKAREA/andre/products/index.html</b></code>
<i>Monospaced bold italics</i>	Used to indicate a variable in response to a system prompt.
[ ]	Square brackets surrounding a command-line argument mean that the argument is optional.
	Vertical bars separating command-line arguments mean that only one of the arguments can be used.

## Editing Text on Windows NT Systems

It is recommended that you use WordPad rather than Notepad to edit text on a Windows NT system. Other text editors may also be used.

## Notation of iw-home on UNIX and Windows NT Systems

### Notation of iw-home on UNIX and Windows Systems

This manual does not use the UNIX notation (*iw-home*; note the lack of italics) except when specifically referring to procedures performed only in UNIX.

This manual uses the Windows version of *iw-home* notation (italicized *iw-home*) when discussing both UNIX and Windows systems. The italics are an Interwoven convention identifying *iw-home* as a variable. You should interpret the *iw-home* notation used in this manual as follows:

- On UNIX systems, *iw-home* is the literal name of the directory containing the TeamSite program files.
- On Windows systems, *iw-home* is the symbolic name of the directory that contains your TeamSite program files. The default value of *iw-home* on Windows systems is:

`C:\Program Files\Interwoven\TeamSite`

The administrator performing the Windows installation may have chosen an installation directory different from the default.

## Chapter 1

# TeamSite Templating Installation

---

TeamSite Templating for end users is available in two versions. *TeamSite Templating Browser* provides a browser-based user interface; all the required software resides on the TeamSite server. *TeamSite Templating Java* provides a Java<sup>®</sup>-based user interface; it requires software be installed on the client machine. A more detailed discussion of these end-user versions is provided in Chapter 3, “Setting Up Data Capture Templates.”

TeamSite must be installed on your server before you can install TeamSite Templating. You must have the same versions of TeamSite and TeamSite Templating. If TeamSite is not installed, see the *TeamSite Administration Guide* for installation instructions. Return to this chapter after TeamSite is installed.

## Hardware Requirements

See the *TeamSite Administration Guide* for general information on hardware requirements.

TeamSite Templating should be installed on a dual CPU server if you plan to enable data content record searches.

Installing TeamSite Templating Java requires an additional 60 megabytes (MB) of hard disk space for the files used by the Java-based interface. On client machines running TeamSite Templating Java, at least 20 MB of hard disk space are required.

## Operating System Requirements

TeamSite Templating is supported on all operating systems that support TeamSite. See the *TeamSite Administration Guide* for information about supported operating systems.

## Pre-Installation Validation

If you are upgrading from TeamSite Templating Classic 4.5 (using a browser-based interface) or from an earlier TeamSite Templating release, you are encouraged to run your data capture templates (DCTs) through an XML validator with the `datacapture5.0.dtd` file before upgrading to TeamSite Templating 5.0. This is necessary because of changes made to the `datacapture5.0.dtd` and the new code that reads the data capture templates. You should also validate your `templating.cfg` file with the templating DTD. A utility command-line tool (CLT) called `iwxml_validate.ipl` is provided with TeamSite Templating to assist you in validating DCTs (see Appendix B, “Command-Line Tools”).

The data capture templates contain validation regex using extended regular expression (regex) syntax. If you have any `datacapture.cfg` files from previous versions that contain validation regex using basic regex(5) syntax, run the `upgrade_dct_cfg.ipl` script on them. This script will convert regex(5) syntax to extended regex syntax. Refer to Appendix B, “Command-Line Tools” for information on this script. You should run this utility once, and only once, on each `datacapture.cfg` file that currently uses basic regex. If you upgraded your `datacapture.cfg` files for TeamSite Templating 4.5, do not attempt to upgrade them again. You cannot convert back to basic regex once you run this utility.

## Installing on Solaris

The TeamSite Templating package for Solaris is available in two forms: a compressed `pkgadd` package stream file or a package directory. If you have downloaded the Templating package, it will be in the compressed package stream form. If you are installing from the CD-ROM, it will be in the package directory form.

To install the package stream package, perform these steps:

1. Log in as **root**.
2. If a previous version of TeamSite Templating was installed, issue the command:  
**# pkgrm IW0Vtst**

3. Unzip and transfer the package stream package into a temporary location by issuing the following command (on one line), where *temp\_dir* is a temporary directory with at least 128 MB of free space:

```
# gunzip < tst.5.5.0.Buildxxxx.pkg.gz | pkgtrans /dev/fd/0 temp_dir IWOVtst
```

4. Install TeamSite Templating by issuing the following command:

```
# pkgadd -d temp_dir IWOVtst
```

5. Remove the temporary directory:

```
# rm -r temp_dir/IWOVtst
```

To install the package directory form, perform these steps:

1. Log in as **root**.
2. Change to the directory containing the IWOVtst directory. If you are installing from CD-ROM, this would be:

```
# cd /cdrom
```

3. Install TeamSite Templating by issuing the following command:

```
# pkgadd -d . IWOVtst
```

Once TeamSite Templating is installed, you must restart the *iwproxy* daemon and the Interwoven Servlet Engine:

1. Log in as **root**.
2. Issue the following command:

```
# iw-home/bin/iwreset -ui
```

See the *TeamSite Administration Guide* for more information about restarting the proxy server.

### Java Runtime Environment Installation Requirements for TeamSite Templating on Solaris

To run the Java runtime environment, you must first install the Solaris operating system patches. You can download these patches from the Sun web site at <http://www.sun.com>.

In Solaris environments, TeamSite can only be run on Sparc or UltraSparc platforms. However, a LaunchPad client is available for the Solaris x86 platform.

### Debugging Information for TeamSite Templating Installation on Solaris

If the installer fails to run on a Solaris server, you can display debugging information by setting the LAX\_DEBUG environment to `true` by entering the following commands at the prompt:

```
% LAX_DEBUG=true ; export LAX_DEBUG
```

```
% sh TSTP.bin
```

## Installing on Windows NT

In Windows NT environments, TeamSite should not be installed on a Primary Domain Controller (PDC) or a Backup Domain Controller (BDC). Additionally, TeamSite can only be installed on a Windows NT server that has been added to a valid Windows NT domain.

Note: When installing a self-extracting NT package, the disk drive where the environment variable TEMP is located must have at least twice as much free space as the package itself in order for the self-extraction to work.

Perform the following steps to install TeamSite Templating on TeamSite running on a Windows NT system:

1. Log on to Windows NT with Administrator permissions.
2. Insert the TeamSite Templating CD into the CD drive. Navigate to the top-level directory and double click the `templating.exe` icon. The Interwoven TeamSite Templating Setup screen appears.
3. Click **Next**. A dialog box appears, prompting for the destination of the TeamSite Templating administrative files. Selecting the default location is recommended; if you specify a new location, it must *not* be the *iw-home* directory.
4. Click **Next**. File names are displayed while the TeamSite Templating administrative files are loaded. A dialog box appears telling you to refer to the *TeamSite Templating Developer's Guide* (this manual).

5. Click **OK**. The TeamSite Templating directory structure shown on page 23 is installed in *iw-home*.
6. Restart the proxy server:
  - Select **Settings > Control Panel** from the **Start** menu.
  - Open the **Services** control panel.
  - Select **Interwoven Proxy** from the list of services.
  - Click **Stop** and wait for service to terminate.
  - Click **Start**. See the *TeamSite Administration Guide* for more information about restarting the proxy server.
7. Restart the Interwoven Servlet Engine.

## Installing on Client Machines

TeamSite Templating Java requires the installation of software on client machines; at least 20 MB of hard disk space are required.

After TeamSite Templating is installed and configured on the server, TeamSite Templating Java is available for users to install on their machines. When content contributors select **File > New Data Record**, they are prompted to install the client-side software. Refer to the *TeamSite Templating User's Guide* for the procedures.

TeamSite Templating Browser does not require the installation of software on client machines, except when VisualFormat is used for formatting text areas. Refer to the *TeamSite Templating User's Guide*.

## Memory Requirements for TeamSite Templating Java Clients

TeamSite Templating client machines running the Java user interface must have at least 64 MB of physical memory.

## Setting Heap Size

If you are using the Java-based interface, the heap size on each client system should be set to one half the physical memory on the machine. To modify the heap size:

1. Stop TeamSite Templating on the client machine.
2. Edit the `tst.lax` file (located in the TeamSite Templating installation directory) to set the heap size. An example of setting the heap size to 64 MB on a 128 MB machine is:

```
lax.nl.java.option.java.heap.size.max=64000000
```

3. Restart TeamSite Templating by selecting a command to create or edit a data content record.

## Multiple Users on a Single Java TeamSite Templating Client

The Java TeamSite Templating client supports multiple users. To have multiple users on a single client machine, each user needs a TeamSite Templating client installation. When installing the TeamSite Templating client, change the location of the files during the second installation to avoid overwriting the first installation.

You also need to make the following changes for each client:

In the `tst.lax` file, set

```
iw.install.port=unique_port_number
```

In the `com_interwoven_templating100_dccredit.properties` file, change the value for `port` to the same port number that was set in the `tst.lax` file as follows:

```
port=unique_port_number
```

## Next Step

After you install TeamSite Templating, you are ready to configure the example templating environment as described in the Chapter 2, “Initial Configuration.”



## Chapter 2

# Initial Configuration

---

After TeamSite Templating is installed, perform the initial configuration described in this chapter. This initial configuration provides a fully functional TeamSite Templating example environment to verify that the TeamSite Templating installation was successful. You can also use the example environment to become familiar with TeamSite Templating features. After you are familiar with the example environment, you can customize it to create your own customized environment as described later in this manual. The configuration activities described in this chapter should be performed by a system administrator or Interwoven consultant.

This chapter begins with an overview of TeamSite Templating configuration, followed by the initial setup activities that will create the example templating environment.

## Configuration Overview

TeamSite Templating provides a highly configurable way to:

- Capture, edit, and store data input from content contributors.
- Define the appearance of displayed data.
- Integrate captured data with other Interwoven features and products such as TeamSite Workflow and DataDeploy.

The TeamSite Templating mechanism for capturing data content from content contributors is separate from the mechanism for defining the appearance of the content when it is displayed. This architecture allows for unlimited reuse of data after the data is captured and stored; it also lets you define different appearances and behaviors for the same data content based on how, when, where, or to whom the data is displayed. You can also use Perl code to extract content from other sources such as relational databases.

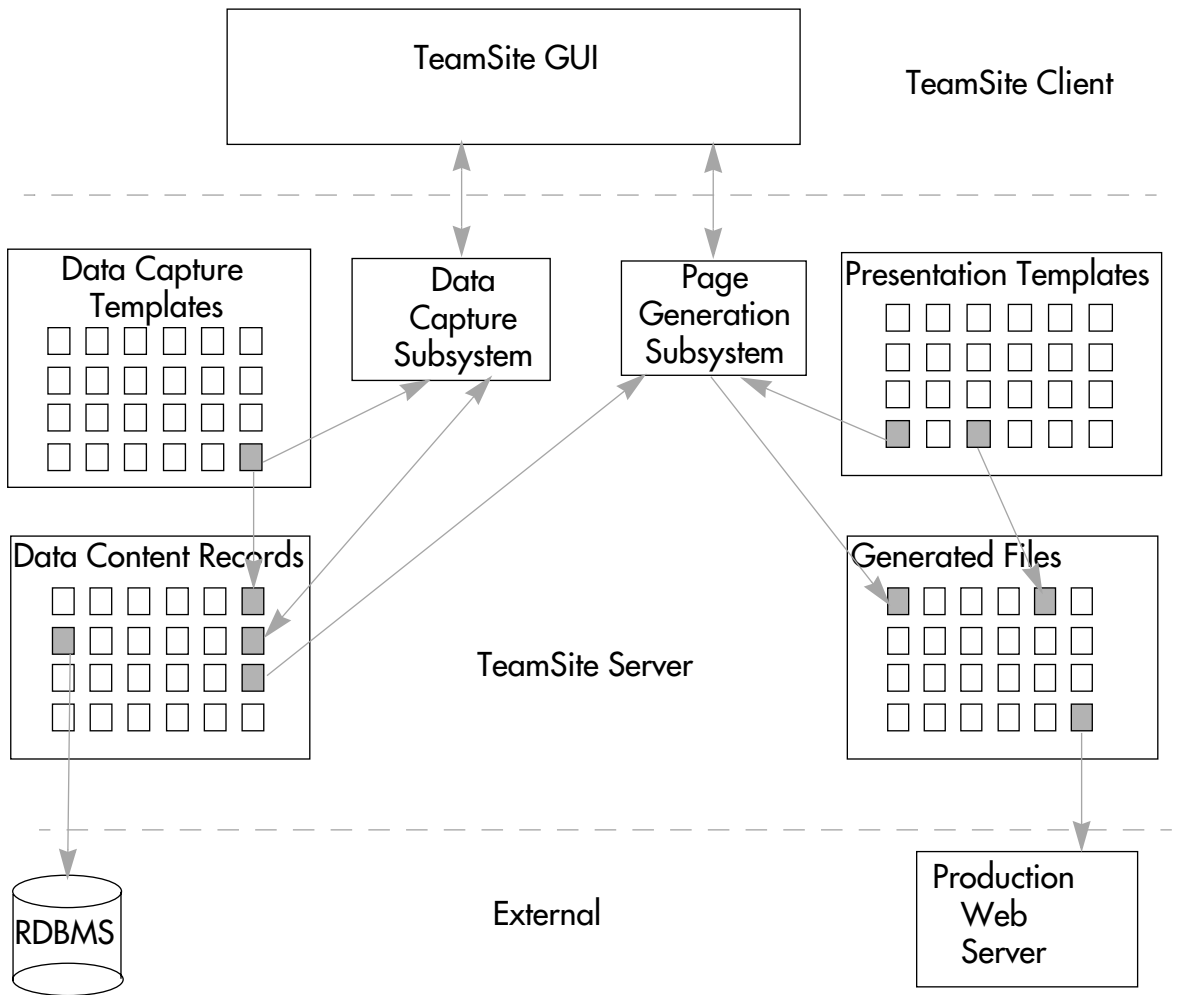
Configuring TeamSite Templating consists of:

- Copying a set of example configuration files and directories supplied with TeamSite Templating into specific locations in your system's directory structure. This sets up a functional example environment that lets you confirm that the TeamSite Templating installation was successful and provides a default environment in which to familiarize yourself with TeamSite Templating. See "Configuring the Example Templating Environment" on page 31 for more information.
- Customizing the templating environment for your specific site by renaming or creating new configuration files (see Chapter 3, "Setting Up Data Capture Templates") and presentation templates (see Chapter 4, "Setting Up Presentation Templates").

## Concepts and Definitions

### TeamSite Templating Model

TeamSite Templating's architecture allows data capture and data presentation to be configured, executed, and managed separately. The following diagram and sections provide a high-level overview of this architecture.



*TeamSite Templating Overview*

## ***Data Capture***

Content contributors working through the TeamSite GUI (either WebDesk or WebDesk Pro) have access to the *data capture subsystem*. This subsystem lets content contributors select and work through forms defined by *data capture templates* to create or edit *data content records*, which by default are stored in the TeamSite file system. Data is stored as XML and used later to fill in *presentation templates* to generate multiple renderings of the content, including for the web and wireless devices. After data content records are created, they can be displayed using presentation templates or optionally deployed to a database using DataDeploy.

The data capture subsystem can be used in either of two interfaces, TeamSite Templating Browser and TeamSite Templating Java.

The interface is determined either for your site or by individual users. Refer to Chapter 3, “Setting Up Data Capture Templates,” for details on configuring the data capture subsystem. Refer to the *TeamSite Templating User’s Guide* for information on using these two interfaces to TeamSite Templating.

## ***Data Presentation***

After data is captured and stored as data content records, users working through WebDesk Pro, WebDesk, or the command line can access the page generation subsystem to combine a data content record with a presentation template. The end result is a generated output file that displays the data content in a way defined by the presentation template. In addition, users can generate an output file that obtains data from multiple data content records and from queries to databases. The generated output file can optionally be deployed to a production web server using OpenDeploy.

## **Definitions**

The following sections define key TeamSite Templating terms.

### ***Data Capture Template***

A *data capture template* is an XML file named `datacapture.cfg` that defines the form used to capture data content from content contributors. A data capture template is associated with a category and type. The category and type define what type of data is required by the data capture template. The data that a content contributor enters in a data capture template is saved on the TeamSite file system as an XML file in the form of a data content record. See “Data Storage Hierarchy” on page 23 for information about where data capture templates reside.

## ***Presentation Template***

A *presentation template* is an XML file that defines how captured data will appear when displayed. A presentation template is populated with a data content record that was captured earlier (through a data capture template) or from queries to databases or by other means such as through the TeamSite TeamCatalog product. You can configure TeamSite Templating to populate any presentation template with any data content record plus any additional information as required from a relational database. You can use presentation templates with component templates. A component template is a nested presentation template that is part of another presentation template. You can also use a single data content record to populate more than one presentation template, resulting in a different look and feel for the same data record. See “Data Storage Hierarchy” on page 23 for information about where presentation templates reside.

## ***Data Content Record***

A *data content record* is an XML file containing formatting information interspersed with data captured from a content contributor or through other means. A data content record is named by the content contributor when it is saved.

## ***Data Capture Subsystem***

The *data capture subsystem* is a set of Java applications that perform the following functions:

- Read the `datacapture.cfg` and `templating.cfg` configuration files to determine what information should be presented to a content contributor using the TeamSite GUI.
- Interpret content contributor input.
- Save content contributor input as formatted data content records.

## ***Page Generation Subsystem***

The *page generation subsystem* is a set of programs and libraries that perform the following functions:

- Read the presentation template and `templating.cfg` configuration files to determine what information should be presented to a content contributor using the TeamSite GUI.
- Interpret content contributor input.
- Combine data content records and presentation templates to produce generated output files.



The presentation template compiler (`iwpt_compile.ipl`) is the primary component of the page generation subsystem. The compiler is a low-level command-line tool that invokes the template parser to create output files. The presentation template compiler is described in more detail in Appendix B, “Command-Line Tools.”

## Configuration Files

TeamSite Templating uses the following configuration files:

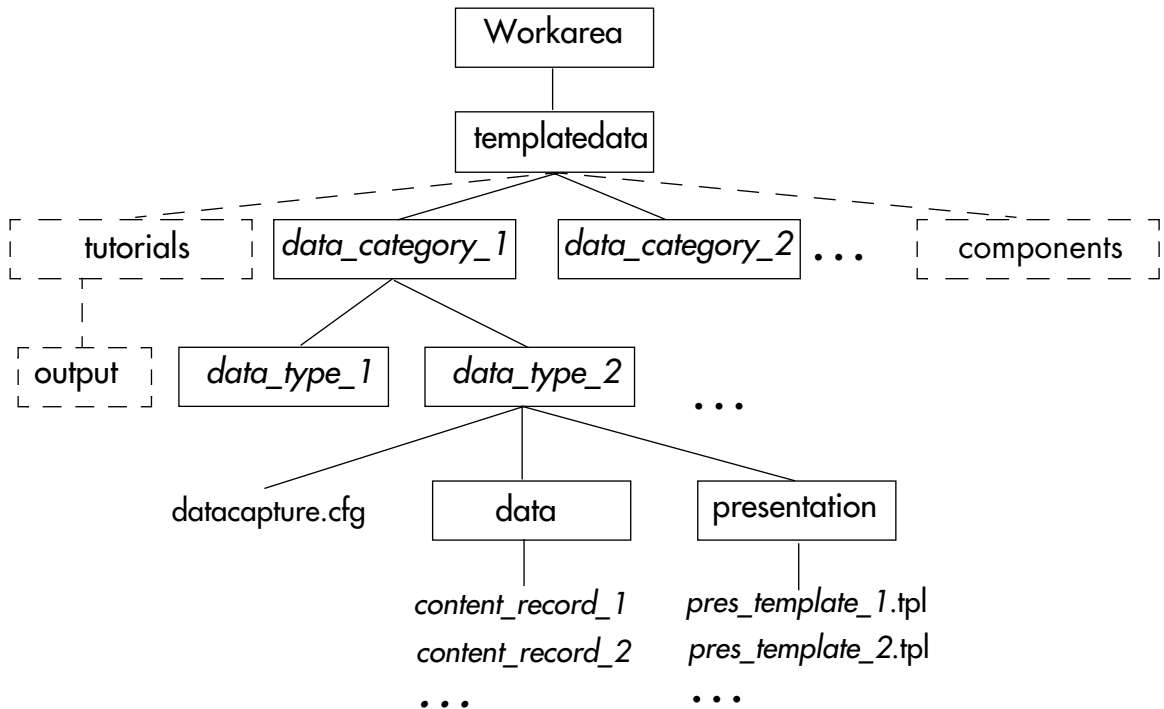
- `templating.cfg`: The main TeamSite Templating configuration file. It is an XML file that resides outside of the TeamSite file system in `iw-home/local/config` and specifies:
  - Which data categories and types are available for use with TeamSite Templating.
  - Which presentation templates can generate HTML files on which TeamSite branches or directories.
  - Which presentation templates can be used with a specific data type.
  - Which users or roles are allowed to create or edit data content records for a specific data type.
  - The location of the presentation template used for previewing generated HTML files.

See Chapter 5, “Mapping Users, Templates, and Content Records,” for details about customizing `templating.cfg`.

- `datacapture.cfg`: An XML file that defines a data capture template and defines the data capture form for a specific data type. It defines the data type itself (such as what information the data type will contain and parameters that define what type of data is legal in any input field). A `datacapture.cfg` file also specifies the look and feel of the data capture form displayed in the TeamSite GUI. Each data type has a `datacapture.cfg` file. A TeamSite Templating environment can contain any number of `datacapture.cfg` files, differentiated by where they reside in the directory structure. See “Data Storage Hierarchy” on page 23 for information about where `datacapture.cfg` files reside. See Chapter 3, “Setting Up Data Capture Templates,” for information about customizing `datacapture.cfg`.
- `available_templates.cfg`: Identifies workflows to be initiated when certain Templating events occur.

## Data Storage Hierarchy

TeamSite Templating uses a data storage hierarchy based on data *categories* and *types*. The directory structure supporting this hierarchy resides in the workarea for each TeamSite Templating user. The directory structure follows. Items in boxes are directories; items not in boxes are files.



*TeamSite Templating Directory Structure*

The `templatedata` directory is at the highest level in the hierarchy.

Data categories are at the next level in the hierarchy and contain one or more data types. For example, the data category `beverages` could contain separate directories for the data types `tea`, `coffee`, `milk`, and so on. In addition to residing in this directory structure, data categories and types must also be listed in the `templating.cfg` configuration file to be made available to TeamSite Templating. See Chapter 5, “Mapping Users, Templates, and Content Records,” for more information. The

components directory that stores component templates and the `tutorials` directory are optional subdirectories of `templatedata`.

Data type directories each contain a `datacapture.cfg` file and the subdirectories `data` and `presentation`. Details for the entire hierarchy are as follows:

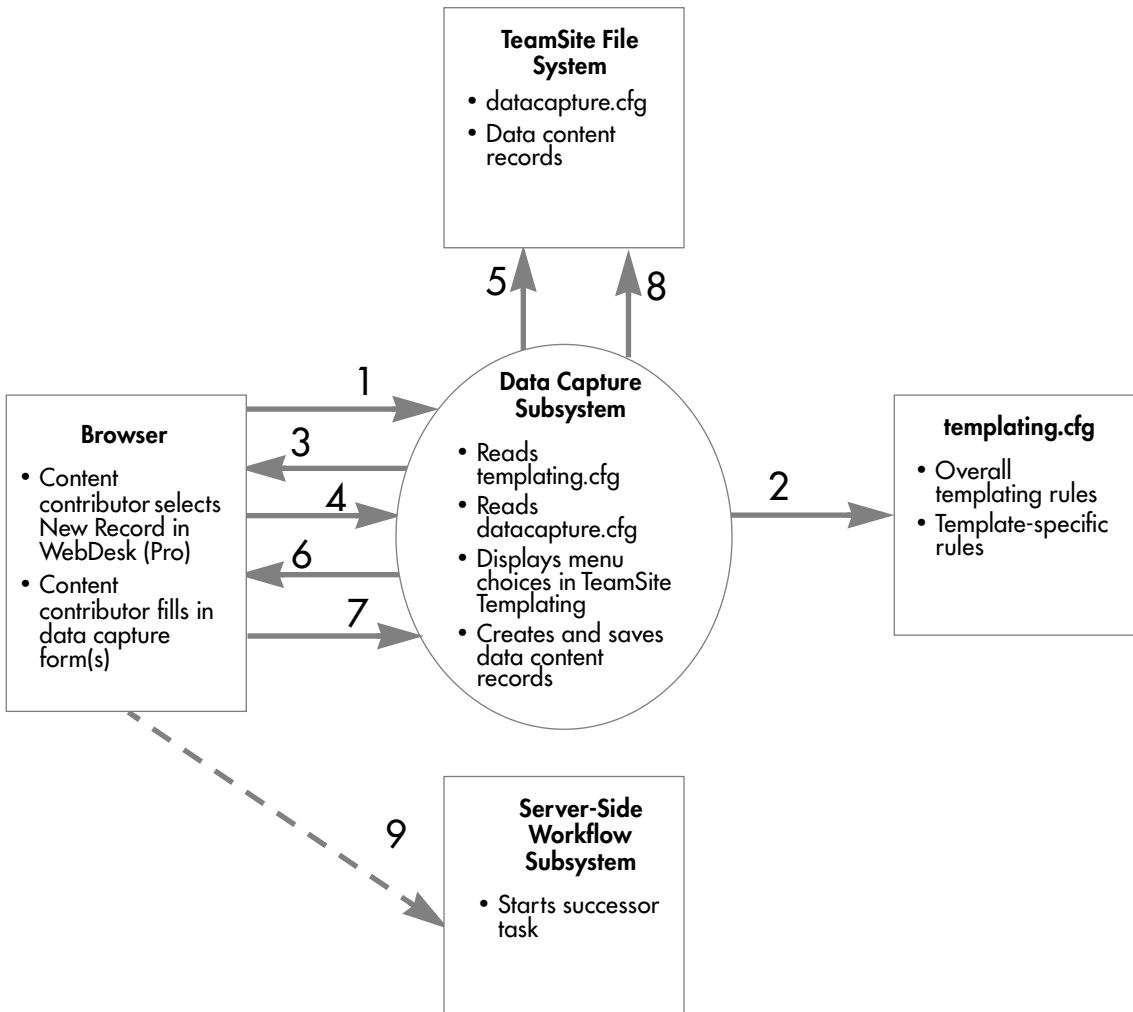
File or Directory	Description
<code>templatedata</code>	Top-level directory containing subdirectories for data categories, types, and all associated configuration files. Resides in the workarea for each user who uses TeamSite Templating. Can be renamed and the <code>iw.cfg</code> file modified.
<code>data_category_1</code>	The first major categorization for data types on a specific branch. Named and defined in <code>templating.cfg</code> . For example: <code>/templatedata/beverages</code>
<code>data_type_1</code>	The first subcategory of data in <code>data_category_1</code> . Named and defined in <code>templating.cfg</code> . For example: <code>/templatedata/beverages/tea</code> . Each data type in a given data category has its own subdirectory.
<code>datacapture.cfg</code>	The XML configuration file that defines a data capture template and drives data capture for a specific data type. It defines the data type itself (such as what information the data type will contain and parameters for what type of data is legal in any input field). Specifies the look and feel of the data capture form displayed in the TeamSite Templating GUI through which a content contributor enters data. Each data type must have exactly one <code>datacapture.cfg</code> file.
<code>data</code>	The directory containing all captured data content records for a given data type. If necessary, you can define and create a directory tree underneath the <code>data</code> directory. A data directory can contain zero or more data content records.
<code>content_record_1</code>	The first data content record for a given data type. Each data content record is an XML file containing formatting information interspersed with data captured from a content contributor using TeamSite Templating. A data content record is named by the content contributor during data entry. For example: <code>/templatedata/beverages/tea/data/november_order</code>



File or Directory	Description
presentation	The directory containing all presentation templates for a given data type. The <code>presentation</code> directory must contain one or more presentation templates.
<i>pres_template_1.tpl</i>	The first presentation template for a given data type. A data type can have any number of presentation templates. A single presentation template is populated by data from one or several data content records and can produce one or several files. A presentation template can have a name of your choice. For example: /templatedata/beverages/tea/presentation/monthly_order.tpl
components	The directory where all component templates are stored. This directory is not required or may be in another location.
tutorials	Examples showing the use of <code>iw_xml</code> tags. This directory is not required or may be in another location.
<i>data_type_2</i>	A second subcategory of data in <i>data_category_1</i> . For example: /templatedata/beverages/coffee
<i>data_category_2</i>	A second major categorization for data on a specific branch. For example: /templatedata/food

## Process Flow: Creating a New Data Content Record

The following diagram shows what takes place when a content contributor creates a new data content record. Sections following the diagram explain each diagram step and component in detail.

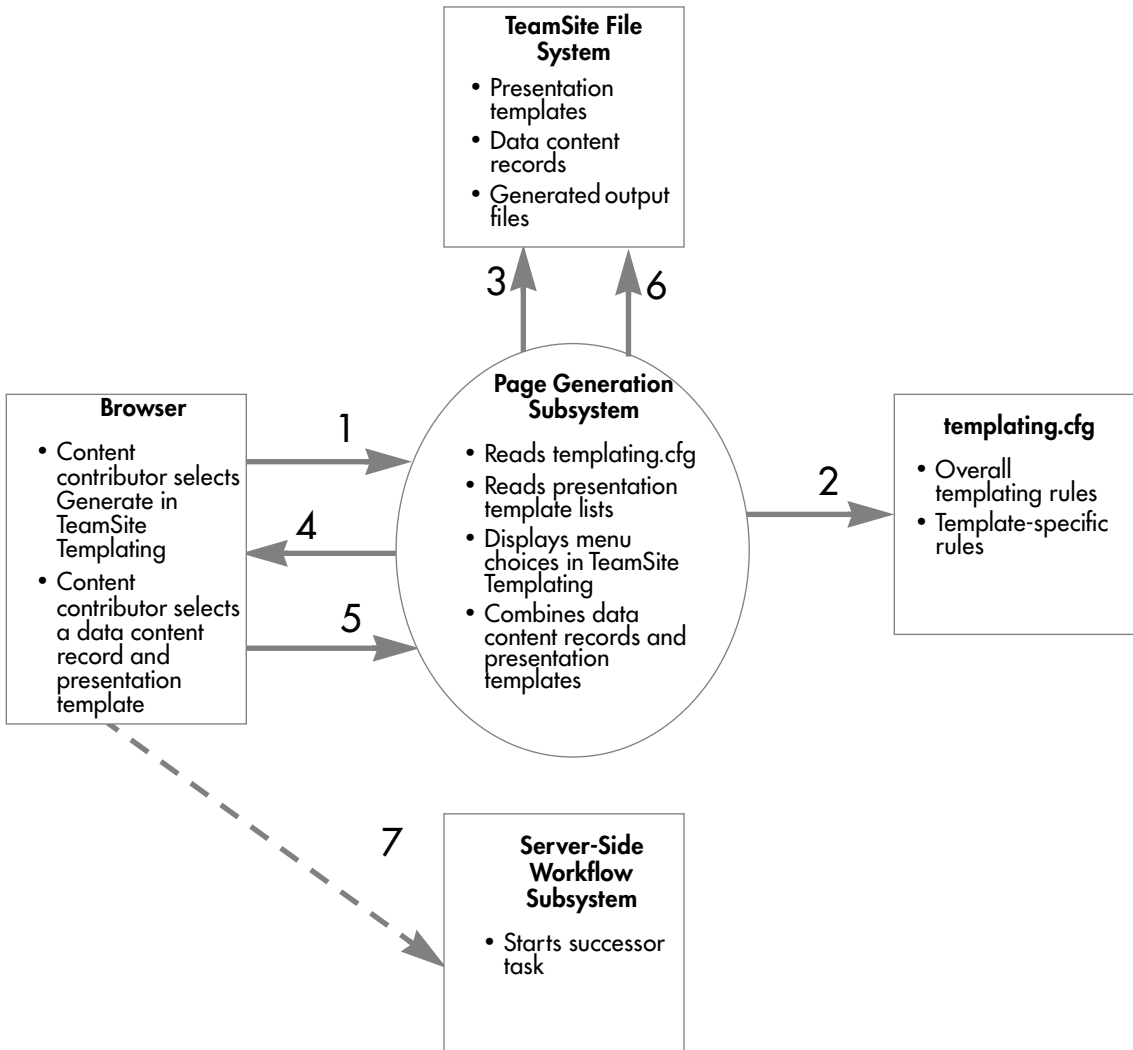


Process Flow Overview: Creating a New Data Content Record

1. A content contributor clicks **File > New Record** in the WebDesk or **File > New Data Record** in WebDesk Pro.
2. The TeamSite Templating data capture subsystem reads the `templating.cfg` file to determine which data types should be displayed as choices for the content contributor. The criteria used for this determination are specified in `templating.cfg` and could include the content contributor's login ID, role, or current TeamSite area or branch. The data type must also exist as a directory in the content contributor's workarea.
3. The data capture subsystem displays the appropriate list of data categories and data types.
4. The content contributor selects a data type. That information is sent back to the data capture subsystem.
5. The data capture subsystem reads the `datacapture.cfg` file for the data type chosen by the content contributor.
6. The data capture subsystem displays the data capture form (as defined by `datacapture.cfg`).
7. The content contributor enters data in the data capture form and selects **File > Save As** to name and save the data content record. The new data is sent to the data capture subsystem.
8. Using the data provided by the content contributor, the data capture subsystem writes a data content record to the TeamSite file system. Note: The content contributor could also have chosen to preview the output file. In that situation, the data capture subsystem reads `templating.cfg` to determine which presentation templates are available for that data type. The content contributor selects a presentation template and the data capture subsystem displays a preview version of the data.
9. If creating the data content record is a task associated with a TeamSite Workflow job, the user indicates the task has been completed and the TeamSite Workflow subsystem starts the successor task.

## Process Flow: Generating an Output File

The following diagram shows the actions that take place when a content contributor generates a new output file by populating a presentation template with a previously captured data content record. Sections following the diagram explain each diagram step and component in detail.



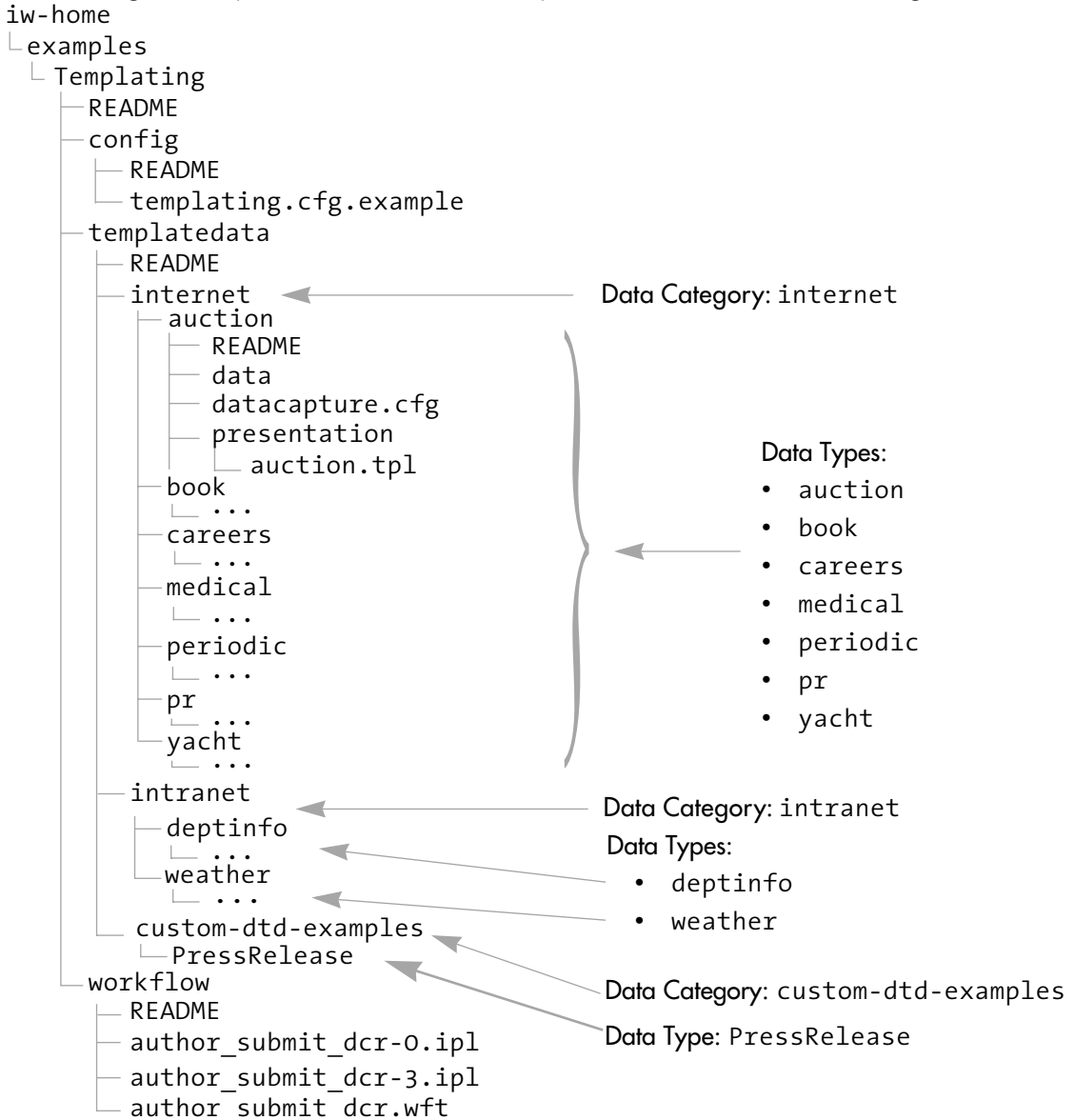
Process Flow Overview: Generating an Output File Using a Data Content Record and a Presentation Template

1. A content contributor clicks **File > Generate Page** from the Java interface or clicks **Generate** from the browser interface.
2. The TeamSite Templating page generation subsystem reads the `templating.cfg` file to determine which data content records for the selected data type should be displayed as choices for the content contributor. The criteria used for this determination are specified in `templating.cfg` and can include the content contributor's login ID, role, or current TeamSite area or branch. The user selects a data content record.
3. The page generation subsystem reads the `/templatedata/data_category/data_type/presentation` directory to determine which presentation templates are associated with the selected data type.
4. The page generation subsystem lists the appropriate presentation templates in the Generate Output window. If only one presentation template is available, the output file displays without the user having to make a selection.
5. The content contributor selects a presentation template.
6. The page generation subsystem generates an output file by entering data from the chosen data content record and any other applicable sources into the chosen presentation template.
7. If creating the generated output file is a task associated with a TeamSite Workflow job, the user indicates that task has been completed and the TeamSite workflow subsystem starts the successor task.



## The Example Directory Structure

The following directory structure is created when you install the TeamSite Templating example:



*TeamSite Templating Example Directory Structure*

The major components of the *iw-home/examples/Templating* directory structure are:

- A top-level README file.
- A config directory containing a README file and an example `templating.cfg` file.
- A `templatedata` directory containing a README file and three data category directories, `internet`, `intranet`, and `custom-dtd-examples`. The `internet` directory contains several data type directories (such as `auction`, `book`, and `pr`). Each data type directory contains a `datacapture.cfg` file, a README file, and the directories `data` and `presentation`. The `presentation` directory for each data type contains at least one presentation template file that generates an output file based on the data content records for that data type. Some data types have multiple presentation templates, in which case any presentation template can be used for output file generation. The `custom-dtd-examples` directory contains an example using the DTD conversion procedures.
- A `workflow` directory containing a README file and all the files necessary to create the workflow job that deploys data content records using DataDeploy. The workflow template `author_submit_dcr.wft` defines the workflow job. The `.ipl` files define external tasks that are components of the job. The workflow job defined by these files executes automatically when an author creates and then submits a data content record to a staging area.

## Configuring the Example Templating Environment

The following sections describe how to configure TeamSite Templating to provide the example templating environment. After the initial setup is complete, you can:

- Use the example templating environment to become familiar with the TeamSite Templating end-user features as described in the *TeamSite Templating User's Guide*.
- Customize the example templating environment as described in the remainder of this manual to create your site-specific configuration.

Perform the following steps to set up the example templating environment. You must copy most of these files and directories to locations that are specific to your site.

1. Decide which workarea you will use for the initial TeamSite Templating setup. Ideally, this workarea should be on a temporary test branch where you can submit and publish without affecting the rest of your TeamSite installation. After TeamSite Templating is configured in the workarea on this test branch, you can copy the workarea to a permanent branch pertaining to your web site. You can then submit the workarea to the staging area and use **Get Latest** to propagate the setup to other workareas on the branch.
2. Read each directory's README file for details about directory contents and last-minute information that might not be documented elsewhere.
3. Copy the following to the specified location, ensuring that all users have read and write permission for each file:

Copy:	To:
<i>iw-home/examples/Templating/templatedata</i>	The workarea determined in Step 1. Copy the entire templatedata directory tree, including the templatedata directory itself. Do not change any directory or file names. The end result should be <i>workarea_name/templatedata/...</i> as shown on page 23.

4. Edit the `available_templates.cfg` file. See “Editing available\_templates.cfg to Initiate Workflows” on page 33.

After you perform these tasks, the example templating environment is fully functional. Workflow files that allow authors to submit data content records were installed in the applicable directories during the TeamSite Templating installation. You can use the example templating environment to create or edit data content records, generate HTML files by combining a data content record with a presentation template, and deploy a data content record's extended attributes to a database using TeamSite workflow and DataDeploy. See Chapter 6, “Integrating Templating, DataDeploy, and Workflow,” for more information about integration with workflow and DataDeploy.



## Editing available\_templates.cfg to Initiate Workflows

The `available_templates.cfg` file contains a series of `<template_file>` elements that specify whether a particular event will be handled by a workflow. Use this file to integrate workflow with templating. You only need to configure `available_templates.cfg` if you want to prompt the user to start or continue with a workflow when a particular event (such as a submit) occurs. Refer to the *TeamSite Workflow Developer's Guide* for details on the syntax for the `available_templates.cfg` file.

To configure `iw-home/local/config/wft/available_templates.cfg`, you must ensure that it contains a `<template_file>` section for each workflow. The following example shows setting up for the Author Submit DCR Workflow.

```
<available_templates>
  <template_file name='Author Submit DCR Workflow'
    path='default/author_submit_dcr.wft'>
    <command_list>
      <command value='tt_data' />
      <command value='tt_deletedcr' />
      <command value='submit' />
      <command value='all' include='no' />
    </command_list>
    <role_list>
      <role value='author' include='yes' allusers='yes' />
      <role value='editor' include='yes' allusers='yes' />
      <role value='all' include='no' />
    </role_list>
  </template_file>
```

This `<template_file>` section says that when authors submit a data content record, they will be prompted to initiate a workflow job using the `author_submit_dcr.wft` workflow. If someone signed on in a role other than author submits a data content record, the workflow will not be initiated. If multiple workflows are available for an action, the author is prompted to select a workflow.



## Modifying the TeamSite iw.cfg File

This section describes some options that may need to be set in the TeamSite `/etc/iw.cfg` file.

### Identifying the Templating Directory

To change the directory in your workareas where templating content will reside, modify the `/etc/iw.cfg` file. The default directory is `templatedata`.

```
[teamsite_templating]
data_root=directory
```

### Identifying the Templating Interface

By default, TeamSite Templating uses the browser-based interface for displaying the data capture form. If you to enable the Java-based interface for use at your site, include the following line in the `/etc/iw.cfg` file. If this option is set to true, users signed on as Editors, Masters, or Administrators may use WebDesk Pro to specify whether they want to use the browser-based or Java-based interface by selecting **View >Change Templating UI**.

```
[teamsite_templating]
use_java_ui=true
```

### Removing the Change Templating UI Menu Item

You may disable the **View >Change Templating UI** menu item by including the following information in the `/etc/iw.cfg` file.

```
[ui_remove_menu_items]
changeTplUI="roleslist"
```

The *roleslist* variable is a comma-separated list of roles for whom the menu item should be disabled (such as `master, editor`).

### Adding DCR Search to the View Menu

If the DataDeploy Database Auto Synchronization has been set up, the data content record search feature is available. Refer to the *DataDeploy Administration Guide* for information on DAS. Uncomment the following line in `/etc/iw.cfg` to add the **Search Data Records** menu item to the TeamSite **View** menu.

```
[iwcgi]
#custom_menu_item_searchdcr="View", "Search Data Records",
"iwsearchdcr.cgi", "all", "scrollbars=yes, resizable=yes,
width=640, height=545", "", "500"
```

Refer to the *TeamSite Administration Guide* for information on metadata capture and search.

### Specifying Search Paths

If the **Search Data Records** custom menu item is made available to users, you can specify the paths that can be searched for data content records. If no `templating_path_regex` statements are specified, all paths are valid. Add a `templating_path_regex` statement in one of the formats shown (the first example is for UNIX platforms; the second example is for Windows platforms). Multiple `templating_path_regex` statements can be specified.

```
[valid_search_paths]
templating_path_regex=^/iwmnt/default/main/www/WORKAREA/w1/
templating_path_regex=^Y:\default\main\www\WORKAREA\w1\
```

If a user attempts a search on an invalid path, a message displays when the user selects **Search Data Records**.

## Providing Content Contributor Interface Access

The Content Contributor interface lets Authors access TeamSite Templating directly from a published intranet page or HTML-formatted email by clicking one or more URL links to the source file. If a user clicks on one of these links and is not already logged in, the TeamSite login screen appears.

If a user has TeamSite Templating installed, you can insert a link to open a data capture record for the specified category and type in the specified workarea. Templating opens in either the Java or browser interface, depending on the system-wide configuration and user preference. Use the syntax:

`http://server/iw/webdesk/newdcr?area_path=pathname&iw_tdt=category_type`

where *server* is the name of the TeamSite server, *pathname* is the area path (such as `/default/main/br/WORKAREA/wa`), and *category\_type* is the Templating data category and type (such as `internet/yacht`).

To insert a link to edit a data content record or an output page, use the syntax:

`http://server/iw/webdesk/edit?vpath=filename`

This link can be hard-coded anywhere you need a link to TeamSite Templating, such as in a Web site, a presentation template, or an HTML-formatted email. See the *TeamSite Administration Guide* for information on other types of links. You can also use the `iwov_webdesk_url` tag to include links in presentation templates. Refer to the Perl doc for this tag for details.

## Starting TeamSite Templating

Perform the following steps to start TeamSite Templating after you have configured the example templating environment:

1. Log out of TeamSite.
2. Log back on to TeamSite.
3. Select **File > New Data Record**.

If a user is configured to use TeamSite Templating Java, a prompt displays to install the client module for the Java interface.

The example templating environment should now be accessible through one of the TeamSite Templating interfaces as described the *TeamSite Templating User's Guide*.

TeamSite Templating can be accessed in the following ways:

- From WebDesk Pro **File** menu.
- From WebDesk **Files** tab.
- From the Smart Content Editing tab.

- From a workflow task (Tasks tab in WebDesk or ToDo list in WebDesk Pro).
- Through the Content Contributor Interface (access Templating by selecting a URL).



## Chapter 3

# Setting Up Data Capture Templates

---

This chapter describes how to edit and create data capture templates. It is assumed that the example templating environment's directory structure exists on your system and that you now intend to customize this environment by creating new data capture templates. See Chapter 2, "Initial Configuration," for more information about the example templating environment's directory structure.

This chapter contains:

- An overview of the user interface and considerations for design.
- An overview of data capture templates.
- Pointers to sample data capture template files that are included with this release of TeamSite Templating.
- A description of the `datacapture.cfg` elements and attributes.
- Examples of data capture forms and the data capture template files that generate the forms.
- A sample data content record.
- The data capture template document type definition (DTD).

You can also create data capture template files from industry-standard XML DTDs. Refer to Appendix C, "Creating DCTs from DTDs."

## User Interface Overview

Two user interfaces are available for TeamSite Templating.

- TeamSite Templating Browser uses a browser-based interface.

- TeamSite Templating Java uses a Java application based interface. It requires that the Java application be installed on the client machine (see the *TeamSite Templating User's Guide* for installation instructions).

Users logged into TeamSite WebDesk Pro can specify the type of interface they want to use by selecting **View > Change Templating UI**. The selection remains in effect until it is changed.

Systems administrators can control the availability of interfaces and make changes to a selected user interface by:

- Enabling or disabling the Java-based interface in `iw.cfg` (page 34).
- Disabling the **View > Change Templating UI** option for specific roles or for all roles (page 34).
- Changing the interface in the entity database using the `iwprop` CLT (see page 134). The entity database stores user credentials as properties for each user.

The primary differences in the two interfaces from the user's viewpoint are in the areas of using replicants and text formatting (see the *TeamSite Templating User's Guide*). The primary differences from the template developer's viewpoint are the method of creating callouts (Appendix A, "Using Callouts"), setting up text formatting options (see page 59), and the ability to configure the appearance of the data capture forms (see page 66).

## Data Capture Template Overview

Data capture templates are XML files named `datacapture.cfg` that reside in the locations described in "Data Storage Hierarchy" on page 23. Each data type is defined with a unique data capture template file (`datacapture.cfg`). Each `datacapture.cfg` file contains the following components:

- *Rule set*: A set of configuration instructions that controls the appearance and behavior of the data capture forms displayed in the TeamSite GUI. A TeamSite Templating `datacapture.cfg` file must contain exactly one rule set. Each rule set contains one or more of the elements identified by the `%items` parameter entity reference (currently `item`, `itemref`, or `container`).



- *Item*: Each item is a single set of data that is to be captured from a content contributor. A rule set must contain at least one item. Items can be nested within other items. If a rule set contains more than one item, item names must be unique within any given nesting level. See page 48 for more information. Each item contains one or more *instances*.
- *Instance*: Each instance defines how to capture data for an item. An instance also defines an ACL that determines which if any instance a specific user is allowed to use to enter the data. See page 50 for more information.

The following list describes the characteristics of data capture forms that you can configure in a `datacapture.cfg` file. Minor customization of the Java-based data capture forms can be made in `dccustomization.xml` (page 66).

- The number and appearance of data capture fields in a data capture form.
- The content and appearance of labels for each data capture field in a data capture form.
- How data will be captured, such as through a check box, radio button, or text field.
- Characteristics of the data entered in each section's fields, including text style, maximum length, and whether the data is text or image.
- Editing capabilities for text areas.
- Which fields, if any, must be filled in before the data entry form can be saved.
- Which fields can be filled in by a specific content contributor.
- Which data entry fields can be displayed multiple times in the same data capture form, and how many times the fields can be displayed.

When a content contributor finishes filling in a data capture form and selects **File > Save**, the data capture subsystem combines the newly entered data with the XML rules defined in the `datacapture.cfg` rule sets. The end result is a data content record that is an XML file that associates field names from the data capture form with the values that were entered in those fields by the content contributor.

Data capture templates should validate against the `datacapture5.0.dtd` file, which can be found at `iw-home/local/config/datacapture5.0.dtd`. However, a validated template is not necessarily a valid file. For example, the `<callout>` element has attributes that are specific or required for one type of callout but not for another.

## Example Data Capture Templates

TeamSite Templating ships with an extensive set of example data capture templates. See “Configuring the Example Templating Environment” on page 31 for descriptions and locations. Two of these templates are described in the remainder of this chapter.

### Data Capture Example 1

The following sections show a hypothetical Press Release data capture form, the `internet/pr/datacapture.cfg` file that generates it, and the data content record that is created when the form is saved.

#### Example 1 Data Capture Form

The following hypothetical Press Release data capture form is included in the TeamSite Templating distribution and is available for use after you configure the example templating environment.

The screenshot shows a web browser window titled "TeamSite Templating". The browser's address bar displays "Press Release - Untitled1". The page content includes a form for entering press release information. The form has a header section with "Type: internet/pr" and "Description: Enter Press Release information.". Below this, there is a date format instruction "Date format is YYYY-MM-DD" followed by a "Publish Date" label and a text input field. A "Headline" label is followed by a text input field. Below the headline is a "Secondary Headline" label and a text input field. An "Introductory Paragraph" label is followed by a large text area and a "VisualFormat..." button. A "Story" section is collapsed, showing a "Subheading" label and a text input field. Below the story section is a "Section Paragraphs" section, which is also collapsed, showing a "Paragraphs" label and a text area with a "VisualFormat..." button. At the bottom of the form, there are three sections: "Enter the name of the author of this article." with an "Author" label and text input field; "Enter your email address." with an "Email" label and text input field; and "Select the language of this article." with a "Language" label and radio buttons for English, German, French, Japanese, Chinese, Spanish, and Italian. The bottom right corner of the browser window shows the "INTE" logo.

TeamSite Templating

File Edit Style Help

Press Release - Untitled1

Type: internet/pr

Description: Enter Press Release information.

Date format is YYYY-MM-DD

Publish Date

Headline

Secondary Headline

Introductory Paragraph  VisualFormat...

Story

Subheading

Section Paragraphs

Paragraphs  VisualFormat...

Enter the name of the author of this article.

Author

Enter your email address.

Email

Select the language of this article.

Language ☐ English ☐ German ☐ French ☐ Japanese ☐ Chinese ☐ Spanish ☐ Italian

INTE

*Press Release Data Capture Form (without data); Displayed using TeamSite Templating Java*

Teamsite Templating Data Capture Form - Microsoft Internet Explorer

File: Untitled Workarea: /default/main/tst/WORK-AREA/tst\_wa

Save Save As Preview Generate Close

**internet/pr**

- Publish Date
- Headline
- Secondary Headline
- Introductory Paragraph
- **Story**
  - Author
  - E-Mail
  - Language

**Description:** Enter Press Release information.

Publish Date\*

Headline\*

Secondary Headline

Introductory Paragraph

**Story-1**

Subheading

**Section Paragraphs-1**

Paragraphs

Author

E-Mail

Language ☐ English ☐ German ☐ French ☐ Japanese ☐ Chinese ☐ Spanish ☐ Italian

*Press Release Data Capture Form (without data); Displayed using TeamSite Templating Browser*

## Example 1 datacapture.cfg File

The datacapture.cfg file that generates this Press Release data capture form is shown below. As with all datacapture.cfg files, it consists of a rule set, items, and instances. See “Explanation of datacapture.cfg File” on page 47 for an explanation of each referenced item. For details about additional datacapture.cfg features not illustrated by this example, see the DTD starting on page 76. An additional sample file specific to TeamSite metadata capture is located in the TeamSite *Administration Guide*. While the syntax for the metadata capture version of datacapture.cfg differs slightly from the TeamSite Templating version, it is similar enough to provide a useful detailed example; refer to that example in addition to the following one.

```
<?xml version="1.0" encoding = "UTF-8"? standalone="no"?>
<!DOCTYPE datacapture SYSTEM "datacapture5.0.dtd">

<data-capture-requirements type="content" name="pr">
  <!-- data-capture-requirements elements contain area elements -->
  <ruleset name="Press Release">
    <description>
      Enter Press Release information.
    </description>

    <!-- file elements contain item elements -->

    <item name="Publish Date">
      <description>Date format is YYYY-MM-DD</description>
      <database data-type="DATE" data-format="yyyy-MM-dd" />
      <text required="t" maxlength="10"
        validation-regex="^[0-9][0-9][0-9][0-9]-[0-1]
          [0-9]-[0-3][0-9]$" />
    </item>

    <item name="Headline">
      <database data-type="VARCHAR(100)" />
      <text required="t" size="50" maxlength="100" />
    </item>
```

DCT Identifier

Rule Set ("Press Release")

Description for the entire data capture template

Item ("Publish Date") with description for item and database elements

Instance (text) with validation regex

Item ("Headline")

Instance (text)



```
<item name="Secondary Headline">
  <database searchable="f" data-type="VARCHAR(200)" />
  <text size="50" maxlength="200" />
</item>
<item name="Introductory Paragraph">
  <database deploy-column="f" />
  <textarea rows="15" cols="64"
    external-editor="visualformat" />
</textarea>
</item>

<item name="Story">
  <database deploy-column="f" />
  <replicant max="4">
    <item name="Subheading">
      <text size="50" maxlength="200" />
    </item>
    <item name="Section Paragraphs">
      <replicant max="4">
        <item name="Paragraphs">
          <textarea rows="15" cols="64"
            external-editor=
              "visualformat" />
        </item>
      </replicant>
    </item>
  </replicant>
</item>

<item name="Author">
  <description>Enter the name of the author
  of this article.</description>
  <database data-type="VARCHAR(40)" />
  <text size="50" maxlength="40" />
</item>
```

Item ("Secondary headline")  
Instance (text)

Item ("Introductory Paragraph")  
Instance (textarea with external editor for formatting)

Item ("Story")  
Instance (replicant)  
Nested Item ("Subheading")  
Nested Instance (text)

Nested Item ("Section Paragraphs")  
Nested Instance (replicant)  
Nested Item ("Paragraphs")  
Nested Instance (textarea with VisualFormat editor specified)

Item ("Author")  
Instance (text)

```

<item name="EMail">
  <description>Enter your email address.
</description>
  <database data-type="VARCHAR(60)" />
  <text maxlength="60" />
</item>

<item name="Languages">
  <description>Select the language of this
    article.</description>
  <database data-type="VARCHAR(10)" />
  <radio>
    <option label="English" value="English" />
    <option label="German" value="German" />
    <option label="French" value="French" />
    <option label="Japanese" value="Japanese" />
    <option label="Chinese" value="Chinese" />
    <option label="Spanish" value="Spanish" />
    <option label="Italian" value="Italian" />
  </radio>
</item>

</ruleset>
</data-capture-requirements>

```

## Explanation of datacapture.cfg File

### DCT Identifier

The `<data-capture-requirements>` element lets you assign a unique identifier for each data capture template. Exactly one `<data-capture-requirements>` element is required in all `datacapture.cfg` files. The name attribute within a `<data-capture-requirements>` element is optional. The type attribute values—content, metadata, and workflow—let you further describe the type of data that will be captured by the template. For data capture templates, content should be specified. The `<dtd-system-identifier>` is a URI indicating the DTD where the `<data-capture-requirements>` were derived. The value of this attribute is used as the system

identifier of the document type declaration when a data content record type is defined as `xml` in the `templating.cfg` file. The information in a `<data-capture-requirements>` element is for reference only. None of the information in this element is stored in the data content record that is created when `datacapture.cfg` is processed by the data capture subsystem.

## Rule Set

The `<ruleset>` element contains the items that make up the rule set that defines the appearance and behavior of the data capture form. A `datacapture.cfg` file must contain exactly one `<ruleset>` element. The name attribute within a `<ruleset>` element is also required. The value of the name attribute appears in the TeamSite GUI as the name of the data capture form (**Press Release** in this example).

Optional subelements are `<label>`, `<description>`, and `(%items;)`. The `<label>` subelement is used to provide a label on the data capture form. The parameter entity reference `(%items;)` is `<item>`, `<itemref>`, or `<container>`. An `<itemref>` is a reference to an item in the `<symbol-table>`. Wherever an `<itemref>` is encountered in the data capture template, the data capture engine replaces it with the appropriate `<symbol>` from the `<symbol-table>` (see Appendix C, “Creating DCTs from DTDs”). This is done dynamically on-demand; for example, if there is an `<itemref>` nested inside an `or` container, it does not get processed until the user gets to it in the GUI. A `<container>` is a non-repeating, named set of data capture items. A `<container>` may appear anywhere in a data capture template that an `<item>` element may appear. A container is conceptually similar to an item with a replicant of `min = 1` and `max = 1`, but it is more efficient. A `<container>` that represents a set of the subelements of an XML element type will contain an `<itemref>` reference for each subelement type it refers to.

## Description

The optional `<description>` subelement inserts a description in the data capture form. A `<description>` subelement can reside anywhere inside the `<ruleset>` element as a child element of `<ruleset>`.

## Item

The `<item>` element assigns a name of your choice to an item and contains the instances and other nested items that specify how to capture data for the item. It also includes information on `rowcontinue` and `colspan`. Use `rowcontinue` to indicate the NEXT item will be placed to the



right of this column; use `colspan` to indicate the item is to be spread over multiple columns. The `rowcontinue` and `colspan` attributes are only supported in TeamSite Templating Browser. A `<ruleset>` element can contain any number of `<item>` elements. Each `<item>` element must contain at least one instance. The optional subelements for `<item>` are `<label>`, `<description>`, and `<database>`. The `<label>` and `<description>` subelements consist of character data. The information provided by `<label>` is used as the field name in the data capture form. If `<label>` is not included, the `name` attribute of the `<item>` element is used as the field name. A `<description>` provides more details about what the data capture item represents or the format that may be required for data entry.

The `<database>` subelement facilitates the use of the appropriate data type in DataDeploy and does not impact templating. The `<database>` subelement has four attributes: `deploy_column` specifies whether a column in the DataDeploy table should be built for that item; `searchable` can be either "t" (default) or "f"; `data-type` is required and is any valid JDBC database type; `data-format` describes the format if date or time is specified for the `data-type` attribute. If a value for `data-format` is specified, the instance should contain a validation regex to force the correct entry in the field.

Item names must be unique within a nested section. For example, the following syntax is *illegal* because it uses the item name `Section` twice in the same nested section in the `<ruleset>` element:

```
<ruleset name="Press Release">
  <item name="Section">
    <text size="40" maxlength="100">
    </text>
  </item>
  <item name="Section">
    <text size="80" maxlength="200">
    </text>
  </item>
</ruleset>
```



However, the following syntax is *legal* because it uses the item name `Section` in different nested sections:

```
<ruleset name="Press Release">
  <item name="Morning Edition">
    <replicant required="t" max="4">
      <item name="Section">
        <text size="80" maxlength="200">
          </text>
        </item>
      </replicant>
    </item>
    <item name="Evening Edition">
      <replicant required="t" max="4">
        <item name="Section">
          <text size="100" maxlength="400">
            </text>
          </item>
        </replicant>
      </item>
    </ruleset>
```

## Instance

An instance defines how to capture data for an item. An instance can also define an ACL that determines which (if any) instance a specific user is allowed to use to enter data. Instances with their attributes and subelements are described in the following table:

Instance	Description
<browser>	<p data-bbox="336 288 1170 322">Lets a content contributor navigate through the workarea to select a file.</p> <p data-bbox="336 331 485 366"><b>Attributes:</b></p> <ul data-bbox="364 374 1280 887" style="list-style-type: none"><li data-bbox="364 374 1280 522">• <b>ceiling-dir</b>: Sets the upper boundary for navigation. The content contributor can never go above the current workarea in the directory structure. The <b>ceiling-dir</b> attribute lets you set the ceiling below the current workarea.</li><li data-bbox="364 531 1280 609">• <b>extns</b>: Comma delimited list of file extensions. Files with these extensions are displayed during navigation.</li><li data-bbox="364 618 1280 687">• <b>initial-dir</b>: The initial directory that is displayed at the start of navigation.</li><li data-bbox="364 696 1280 730">• <b>size</b>: The number of characters that can display in the browse field.</li><li data-bbox="364 739 1280 774">• <b>maxlength</b>: The maximum number of characters the user can enter.</li><li data-bbox="364 782 1280 887">• <b>required</b>: Specifies whether data must be captured by this instance. The default setting is <b>f</b> (not required). Setting it to <b>t</b> specifies that a user must specify a value for this item.</li></ul>



Instance	Description
<code>&lt;browser&gt;</code> (continued)	<p><b>Subelements:</b></p> <ul style="list-style-type: none"><li>• <code>&lt;allowed&gt;</code>: Lets you set an ACL to specify which users can or cannot use a specific instance to enter data. If <code>&lt;allowed&gt;</code> is not set, any user can enter data for the instance. The <code>&lt;allowed&gt;</code> element can have any of the following subelements:<ul style="list-style-type: none"><li>– <code>&lt;cred&gt;</code>: Lets you name a user or role in the ACL (for example, <code>user="joe"</code> or <code>role="master"</code>).</li><li>– <code>&lt;and&gt;</code>: Logical and statement for grouping ACL credentials.</li><li>– <code>&lt;or&gt;</code>: Logical or statement for grouping ACL credentials.</li><li>– <code>&lt;not&gt;</code>: Logical not statement for negating ACL credentials. Users who are not allowed do not see the instance on their data capture form.</li></ul></li></ul> <p>See the examples on page 60.</p> <ul style="list-style-type: none"><li>• <code>&lt;callout&gt;</code>: Creates a button that calls an external program. This subelement is maintained for compatibility with previous versions and should not be used in new data capture files.<ul style="list-style-type: none"><li>– <code>type</code>: Must be <code>java-class</code> or <code>cgi</code>.</li><li>– <code>location</code>: Specifies the URL of a jar file or class file. The file does not necessarily have to be on the same server as TeamSite Templating.</li><li>– <code>class</code>: Specifies the actual name of the class in the jar file.</li><li>– <code>label</code>: Identifies the text on the button in the data capture form.</li><li>– <code>window-features</code>: Determines how the window with the CGI program displays. See the discussion of <code>WindowAttributes</code> in the <i>TeamSite Administration Guide</i>.</li></ul></li></ul>

Instance	Description
<browser> (continued)	<p><b>Subelements (continued):</b></p> <ul style="list-style-type: none"> <li>• <b>&lt;java-callout&gt;</b>: Creates a button that calls a Java external program for use with TeamSite Templating Java (see Appendix A, “Using Callouts”). It has the following elements: <ul style="list-style-type: none"> <li>– <b>location</b>: Specifies the URL of a jar file or class file. The file does not necessarily have to be on the same server as TeamSite Templating. Supported only in TeamSite Templating Java.</li> <li>– <b>class</b>: Specifies the actual name of the class in the jar file.</li> <li>– <b>label</b>: Label of the button that launches the callout code.</li> </ul> </li> <li>• <b>&lt;cgi-callout&gt;</b>: Creates a button that calls a CGI program for use with TeamSite Templating Browser (see Appendix A, “Using Callouts”). It has the following elements: <ul style="list-style-type: none"> <li>– <b>url</b>: URL of the CGI program to run. Supported only in TeamSite Templating Browser.</li> <li>– <b>label</b>: Identifies the text on the button in the data capture form.</li> <li>– <b>window-features</b>: Determines how the window with the CGI program displays. See the discussion of WindowAttributes in the <i>TeamSite Administration Guide</i>.</li> </ul> </li> </ul>

Instance	Description
<checkbox>	<p>Specifies that data will be captured using one or more check boxes.</p> <p><b>Attributes:</b></p> <ul style="list-style-type: none"> <li>• <b>delimiter:</b> Specifies the delimiting character used when data from all check boxes is concatenated by the data capture subsystem. The default delimiter is a comma (,).</li> <li>• <b>required:</b> See &lt;browser&gt;.</li> </ul> <p><b>Subelements:</b></p> <ul style="list-style-type: none"> <li>• <b>&lt;allowed&gt;:</b> See &lt;browser&gt;.</li> <li>• <b>&lt;callout&gt;:</b> See &lt;browser&gt;.</li> <li>• <b>&lt;java-callout&gt;:</b> See &lt;browser&gt;.</li> <li>• <b>&lt;cgi-callout&gt;:</b> See &lt;browser&gt;.</li> <li>• <b>(%chooser-options;):</b> Parameter entity reference that has the following values: <ul style="list-style-type: none"> <li>– <b>&lt;inline&gt;:</b> Provides a method for making server-side inline callout programs that return multiple XML elements to the data capture form (see page 64 for additional details).</li> <li>– <b>&lt;option&gt;:</b> Lets you assign a label or value to a check box so that a user can enter only the predetermined label or value data by checking the check box. Also lets you specify whether the check box is initially displayed as being selected by default. A &lt;checkbox&gt; element must have at least one &lt;option&gt; subelement. See the DTD on page 76 for syntax details.</li> </ul> </li> </ul>

Instance	Description
<hidden>	<p>Specifies that the data will not be shown in the data capture form. A &lt;hidden&gt; field may receive data from a callout program. Note: You cannot currently specify a default value.</p> <p><b>Attributes:</b></p> <ul style="list-style-type: none"> <li>• required: See &lt;browser&gt;.</li> </ul> <p><b>Subelements:</b></p> <ul style="list-style-type: none"> <li>• &lt;allowed&gt;: See &lt;browser&gt;.</li> <li>• &lt;callout&gt;: See &lt;browser&gt;.</li> <li>• &lt;java-callout&gt;: See &lt;browser&gt;.</li> <li>• &lt;cgi-callout&gt;: See &lt;browser&gt;.</li> </ul>
<radio>	<p>Specifies that data will be captured using one or more radio buttons. Note: You cannot currently specify a default value.</p> <p><b>Attributes:</b></p> <ul style="list-style-type: none"> <li>• required: See &lt;browser&gt;.</li> </ul> <p><b>Subelements:</b></p> <ul style="list-style-type: none"> <li>• &lt;allowed&gt;: See &lt;browser&gt;.</li> <li>• &lt;callout&gt;: See &lt;browser&gt;.</li> <li>• &lt;java-callout&gt;: See &lt;browser&gt;.</li> <li>• &lt;cgi-callout&gt;: See &lt;browser&gt;.</li> <li>• &lt;inline&gt;: See &lt;checkbox&gt;.</li> <li>• &lt;option&gt;: See &lt;checkbox&gt;. A &lt;radio&gt; element must have at least one &lt;option&gt; subelement.</li> </ul>
<readonly>	<p>Specifies that the data will be shown on the data capture form but will not be editable.</p> <p><b>Subelements:</b></p> <ul style="list-style-type: none"> <li>• &lt;allowed&gt;: See &lt;browser&gt;.</li> <li>• &lt;callout&gt;: See &lt;browser&gt;.</li> <li>• &lt;java-callout&gt;: See &lt;browser&gt;.</li> <li>• &lt;cgi-callout&gt;: See &lt;browser&gt;.</li> </ul>



Instance	Description
<replicant>	<p>Specifies a repeatable instance that can contain multiple nested items and instances. When there are multiple instances, the first instance whose ACL allows the current user to enter data will be the instance used for that user. &lt;replicant&gt; is the only instance that can contain nested items and instances. Whenever additional iterations of the instance can be displayed (that is, if the <b>max</b> threshold has not yet been reached), the <b>Insert</b> menu item or icon is active. Whenever iterations of the instance can be removed (that is, if the <b>min</b> threshold has not yet been reached), the <b>Delete</b> menu item or icon is active. If a &lt;replicant&gt; has four items, the <b>Insert</b> menu item displays another set of four items in the data capture form.</p> <p><b>Attributes:</b></p> <ul style="list-style-type: none"><li>• <b>default</b>: The number of instance iterations displayed initially in the data capture form.</li><li>• <b>max</b>: The maximum number of items that can reside within the replicant instance.</li><li>• <b>min</b>: The minimum number of items that can reside within the replicant instance.</li><li>• <b>combination</b>: Specifies whether the entire set of items will be replicated when the user requests a replicant or whether the user will be prompted to select one of the replicant items. The valid values are <b>or</b> and <b>and</b>.</li><li>• <b>hide-name</b>: Determines whether the label displays for each replicant.</li></ul> <p><b>Subelements:</b></p> <ul style="list-style-type: none"><li>• <b>&lt;allowed&gt;</b>: See <b>&lt;browser&gt;</b>.</li><li>• <b>&lt;item&gt;</b>: See <b>Item</b> on page 48.</li><li>• <b>&lt;itemref&gt;</b>: A &lt;container&gt; that represents a set of the subelements of an XML element type will contain an &lt;itemref&gt; reference for each subelement type it refers to.</li><li>• <b>&lt;container&gt;</b>: A &lt;container&gt; is a non-repeating, named set of data capture items. In addition to the <b>combination</b> and <b>hide-name</b> attributes, you can also include the <b>name</b> attribute to specify the field name.</li></ul>



Instance	Description
<select>	<p>Specifies that data will be captured using a drop-down list.</p> <p><b>Attributes:</b></p> <ul style="list-style-type: none"> <li>• <code>delimiter</code>: See &lt;checkbox&gt;.</li> <li>• <code>required</code>: See &lt;browser&gt;.</li> <li>• <code>multiple</code>: Specifies whether more than one item can be selected. The default value is <code>f</code> (only one item can be selected). Setting <code>multiple="t"</code> specifies that a user can select more than one item.</li> <li>• <code>size</code>: The number of selections that display in the selection box at one time when <code>multiple</code> is <code>t</code>.</li> <li>• <code>width</code>: The width of the drop-down or select list. (Not respected by browser UI.)</li> </ul> <p><b>Subelements:</b></p> <ul style="list-style-type: none"> <li>• <code>&lt;allowed&gt;</code>: See &lt;browser&gt;.</li> <li>• <code>&lt;callout&gt;</code>: See &lt;browser&gt;.</li> <li>• <code>&lt;java-callout&gt;</code>: See &lt;browser&gt;.</li> <li>• <code>&lt;cgi-callout&gt;</code>: See &lt;browser&gt;.</li> <li>• <code>&lt;inline&gt;</code>: See &lt;checkbox&gt;.</li> <li>• <code>&lt;option&gt;</code>: See &lt;checkbox&gt;.</li> </ul>

Instance	Description
<code>&lt;text&gt;</code>	<p>Specifies that data will be entered and captured using an unformatted single-line text field.</p> <p><b>Attributes:</b></p> <ul style="list-style-type: none"><li>• <code>maxlength</code>: The maximum number of characters the user can enter.</li><li>• <code>required</code>: See <code>&lt;browser&gt;</code>.</li><li>• <code>size</code>: The number of characters that display in the text box.</li><li>• <code>validation-regex</code>: Uses extended regex syntax to set validation criteria for text entered by a user. A retry message appears in the data capture form if the entered text does not meet the specified criteria.</li></ul> <p><b>Subelements:</b></p> <ul style="list-style-type: none"><li>• <code>&lt;allowed&gt;</code>: See <code>&lt;browser&gt;</code>.</li><li>• <code>&lt;default&gt;</code>: The default text that displays in the field when the data capture form opens.</li><li>• <code>&lt;callout&gt;</code>: See <code>&lt;browser&gt;</code>.</li><li>• <code>&lt;java-callout&gt;</code>: See <code>&lt;browser&gt;</code>.</li><li>• <code>&lt;cgi-callout&gt;</code>: See <code>&lt;browser&gt;</code>.</li></ul>

Instance	Description
<textarea>	<p>Specifies that data will be entered and captured in a text field of a specified size.</p> <p><b>Attributes:</b></p> <ul style="list-style-type: none"> <li>• <b>cols:</b> The width (in characters) of the text area. If <code>rtf="t"</code> is set, width in pixels.</li> <li>• <b>required:</b> See &lt;browser&gt;.</li> <li>• <b>rows:</b> The height (in rows) of the text area. If <code>rtf="t"</code> is set, height in pixels.</li> <li>• <b>wrap:</b> Handles word wrapping in text input areas in forms. When <code>off</code> is set, lines are sent exactly as typed; when <code>virtual</code> is specified, the text word wraps in the form, but long lines are sent as one line; when <code>physical</code> is set, the word wraps and text are transmitted at all wrap points. Note: The Java UI does not distinguish between <code>virtual</code> and <code>physical</code>.</li> <li>• <b>validation-regex:</b> See &lt;text&gt;. The <code>validation-regex</code> and <code>rtf</code> attributes cannot both be specified for the same textarea.</li> <li>• <b>rtf:</b> Allows user to set text styles of bold, italics, underscoring, and hyperlink for Java-based templates.</li> <li>• <b>line-break:</b> Controls how line breaks are handled when <code>rtf="t"</code>. If set to <code>P</code>, all hard returns in the <code>rtf</code> content are replaced by surrounding <code>&lt;P&gt;</code> and <code>&lt;/P&gt;</code> tags; that is, each line is converted into a HTML equivalent of a paragraph. If set to <code>BR</code>, all hard returns are replaced by <code>&lt;BR&gt;</code>. If a line-break is not specified, hard returns are preserved as they are in the content. This setting is not valid for the VisualFormat editor.</li> <li>• <b>external-editor:</b> Specifies the editor to be used for text formatting. The currently accepted value is "VisualFormat". This option is only valid for clients on Windows platforms (both Java and browser interfaces).</li> <li>• <b>external-editor-config:</b> Specifies which configuration files can be used. The default file is <code>/iw/config/visualformatconfig.xml</code>. If the path starts with a <code>/</code>, it is assumed to be an absolute path on the TeamSite Web server. Otherwise, the file is assumed to be in <code>/iw/config</code>. Refer to the <i>TeamSite Templating: VisualFormat Developer's Guide</i>.</li> </ul>

Instance	Description
<code>&lt;textarea&gt;</code> (continued)	<b>Subelements:</b> <ul style="list-style-type: none"> <li>• <code>&lt;allowed&gt;</code>: See <code>&lt;browser&gt;</code>.</li> <li>• <code>&lt;default&gt;</code>: See <code>&lt;text&gt;</code>.</li> <li>• <code>&lt;callout&gt;</code>: See <code>&lt;browser&gt;</code>.</li> <li>• <code>&lt;java-callout&gt;</code>: See <code>&lt;browser&gt;</code>.</li> <li>• <code>&lt;cgi-callout&gt;</code>: See <code>&lt;browser&gt;</code>.</li> </ul>

### Details on Attributes and Subelements of Instances

This section provides additional details on the attributes and subelements described in the instance table.

#### The `<allowed>` Attribute

The following code allows all users except joe to use the current instance:

```
<allowed>
  <not>
    <cred user="joe">
    </cred>
  </not>
</allowed>
```

In the following example, `<allowed>` sets one instance for editors and another instance for all other roles. The first instance a user satisfies is the one that is used.

```
<item name= "abc">
  <instance>
    <!--only for editors-->
    <allowed> <cred role="editor"/> </allowed>
  </instance>
  <instance>
    <!--for everyone else-->
  </instance>
</item>
```

The following example shows how to prohibit multiple roles from editing a field. If the `<or>` statement is not included, only the first role is not allowed.

```
<allowed>
  <not>
    <or>
      <cred role='master' />
      <cred role='admin' />
      <cred role='editor' />
      <cred role='author' />
    </or>
  </not>
</allowed>
```

The following code segments show some examples of using `<allowed>` with `<readonly>`. The `<readonly>` tag makes a field *read-only* to certain users, which means that they can only read it, and they cannot write to it.

This example says that only the user `chris` gets a text field for the **Headline**. Implicitly, it is read-only for everybody else.

```
<item name="Headline">
  <text>
    <allowed><cred user="chris"/></allowed>
  </text>
</item>
```

The functionality of the preceding example is identical to that of the following example that includes the `<readonly>` element:

```
<item name="Headline">
  <text>
    <allowed><cred user="chris"/></allowed>
  </text>
  <readonly/>
  <!-- Absence of an <allowed> tag means "everyone". -->
</item>
```



You could reverse it so the user `chris` gets a read-only field, and everyone else gets a text box:

```
<item name="Headline">
  <readonly>
    <allowed><cred user="chris"/></allowed>
  </readonly>
  <text/>
</item>
```

### The `<required>` and `<hidden>` Attributes

The following code limits access to a required text area. Master users can see the text area and must fill it out, but nobody else can see the field. Users who are not allowed to see this text area will not be forced to fill it out and will be able to save the data content record even though the value for the text area is not set.

```
<item name="SecretComments">
  <textarea cols="15" rows="10" required="t">
    <allowed>
      <cred role="master" />
    </allowed>
  </textarea>
  <hidden/>
</item>
```

The following code also limits access to a required text area. In this case, everyone except Master users can see the text area and must fill it out. Master users will be able to save the data content record even though the value is not set.

```
<item name="SecretComments">
  <hidden >
    <allowed>
      <cred role="master" />
    </allowed>
  </hidden>
  <textarea cols="15" rows="10" required="t" />
</item>
```

The following code requires an Author to enter a value for a hidden field (for example, through a callout button). Masters, Administrators, and Editors may enter a value but are not required to.

**Note:** Use this functionality with extreme caution. Requiring users to enter information when they cannot see that the information is required can severely impair usability.

```
<item name="SecretComments">
  <hidden required="t" >
    <allowed>
      <cred role="author" />
    </allowed>
  </hidden>
  <textarea cols="15" rows="10" / >
</item>
```

In the following code, all users are required to have a value in the text area. However, the text area is hidden for Authors. In this scenario, an Editor, Administrator, or Master user would create the data content record and set a hidden field, and only then would Authors be able to edit and save the data content record. Authors cannot save this data content record if the hidden field has not been pre-filled.

**Note:** Use this functionality with extreme caution. Requiring users to enter information when they cannot see that the information is required can severely impair usability.

```
<item name="SecretComments">
  <hidden required="t" >
    <allowed>
      <cred role="author" />
    </allowed>
  </hidden>
  <textarea cols="15" rows="10" required="t" / >
</item>
```

### The <rtf> and <external-editor> Attributes

The <rtf> attribute for a <textarea> element allows users of Java templates to set text styles of bold, italics, and underscore. It also provides an easy method of setting a hyperlink. The <line-break> attribute controls how line breaks are handled in textareas that have rtf set.

The <external-editor> attribute for a <textarea> element provides VisualFormat editing capabilities for both browser-based and Java-based templating users. However, users must be using a client machine running a Windows operating system. Refer to the *TeamSite Templating User's Guide* for information on using this feature.

Template developers may specify a configuration file to be used for a specific text area. Refer to the *TeamSite Templating:VisualFormat Developer's Guide* for information on creating configuration files.

The following table identifies the editing and formatting capabilities on various client platforms.

	<b>Browser-based Templating</b>	<b>Java-based Templating</b>
Windows platforms	VisualFormat editing available	RTF formatting available VisualFormat editing available If both are specified, VisualFormat takes precedence.
Unix platforms	No formatting capability	RTF formatting available
Macintosh platforms	No formatting capability	RTF formatting available

### The <inline> Subelement

The <inline> element is shown in the DTD for the <checkbox>, <radio>, and <select> elements. You can actually use it anywhere in the data capture template but probably are most likely to use it in these three elements. The <inline> element cannot contain <cred> subelements or additional <inline> elements.

An <inline> element should have a command attribute such as:

```
<inline command="/bin/cat /tmp/a /tmp/b"/>
```



The inline callout program should return a well-formed XML document. The document's outermost element should be a `<substitution>` element. It should contain any XML that is valid according to `datacapture5.0.dtd`. That `<substitution>` element will contain six `<option>` elements, enumerating a variety of types of yacht hull materials (see page 72).

```
<?xml version="1.0" encoding="UTF-8"?>
<substitution>
  <option value="Lead" label="Lead"/>
  <option value="Tin" label="Tin"/>
  <option value="Silicon" label="Silicon"/>
  <option value="Plastic" label="Plastic"/>
  <option value="Paper" label="Paper"/>
  <option value="Glass" label="Glass"/>
</substitution>
```

This simple callout outputs a static result. A more sophisticated callout program could query a database and return the query results as `<option>` elements.

When a server-side inline callout program is executed, it inherits the following environment variables:

- `IW_DCT`: The file system path to the `datacapture.cfg` in use (for example, `/iwmnt/default/main/development/WORKAREA/maudlin/templatedata/press/events/datacapture.cfg`).
- `IW_ROLE`: The role of the current data capture user (for example, `editor`).
- `IW_USER`: The full user name of the current templating user (for example, `andre`).
- `IW_WORKAREA`: The vpath to the current workarea (for example, `/default/main/development/WORKAREA/chris`).

The exit status from a server-side inline callout should be 0 to indicate a successful execution. Normally an inline callout should not return a non-zero value. However, an example where a non-zero value may be needed to indicate an error condition is if you are populating a `<select>` menu by making a database query and the database is offline. Rather than displaying a form with no choices, you may prefer an exception be displayed.



## Customizing the Appearance of Java Data Capture Forms

To customize the appearance of your Java-based data capture forms use the `dccustomization.xml` file. This file is located on the TeamSite server at `iw-home/httpd/iw/config/dccustomization.xml`. The default file is:

```
<templating-ui-customization>
  <datacapture-ui-customization>
    <form bgcolor="#FFFFFF"/>
    <required-field>
      <label color="#BB0000" font="Dialog" fontstyle="bold"/>
    </required-field>
    <field>
      <label color="#000000" font="Dialog" fontstyle="plain"/>
    </field>
    <logo name="newtslogo.gif"/>
  </datacapture-ui-customization>
</templating-ui-customization>
```

Colors are specified by their hexadecimal value, preceded by a `#` sign (similar to specifying colors in HTML files). Tags are described as follows:

Tag	Description								
form bgcolor	The color of the data capture form.								
required field	The labels for all required fields in the form. You may specify a label color, font, and font style.								
field	The labels for all non-required fields in the form. You may specify a label color, font, and font style.								
label color	The color of the label used to identify the field or required field. Specifying different label colors for the two field types assists users who are entering data in the data capture forms.								
font	One of the following logical font names must be specified. The logical font name is mapped to a native font for the underlying windowing system. If the specified font name is not recognized by the system, the default font is used. The font names are: <table><tr><td>Dialog</td><td>Helvetica</td></tr><tr><td>SansSerif</td><td>TimesRoman</td></tr><tr><td>Serif</td><td>Courier</td></tr><tr><td>Monospaced</td><td></td></tr></table>	Dialog	Helvetica	SansSerif	TimesRoman	Serif	Courier	Monospaced	
Dialog	Helvetica								
SansSerif	TimesRoman								
Serif	Courier								
Monospaced									
fontstyle	The supported font styles are: <table><tr><td>bold</td><td>italic</td><td>plain</td></tr></table>	bold	italic	plain					
bold	italic	plain							
logo	The logo in the upper left corner of the data capture form can be changed by entering a .gif file name. The .gif file must be in the <i>iw-home/httpd/iw-icons</i> directory.								

One file is used for all data categories and types. Templating clients must restart TeamSite Templating Java for changes to take effect.

## Example 1 Data Content Record

This section shows the data content record that is created if a content contributor enters the following data in the Press Release data capture form:

**Teamsite Templating Data Capture Form - Microsoft Internet Explorer**

File: templatedata/internet/pr/data/pr1 Workarea: /default/main/cb1/WORKAREA/cw1

Save Save As Preview Generate Close

**internet/pr**

- Publish Date
- Headline
- Secondary Headline
- Introductory Paragraph
- ▼ **Story**
  - ▼ **Story-1**
    - Subheading
    - ▼ **Section Paragraphs**
      - ▼ **Section Paragraphs-1**
        - Paragraphs
  - Author
  - EMail
  - Language

**Description:** No description is available for this item.

Publish Date\* 2001-04-24

Headline\* Candidate Joins Race

Secondary Headline

Introductory Paragraph

**Story-1**

Subheading

**Section Paragraphs-1**

Paragraphs

A new candidate enters the race as of 4/24/01.

Author eal

EMail eal@example.com

Language ☒ English ☐ German ☐ French ☐ Japanese ☐ Chinese ☐ Spanish ☐ Italian

Press Release Data Capture Form (with data)

The resulting data content record is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE record SYSTEM "dcr5.0.dtd">
<record name="eal.pr.1" type="content">
  <item name="Publish Date">
    <value>2001-04-24</value>
  </item>
  <item name="Headline">
    <value>Candidate Joins Race</value>
  </item>
  <item name="Secondary Headline">
    <value></value>
  </item>
  <item name="Introductory Paragraph">
    <value></value>
  </item>
  <item delimiter=", " name="Story">
    <value>
      <item name="Subheading">
        <value></value>
      </item>
      <item name="Section Paragraphs">
        <value>
          <item name="Paragraphs">
            <value>A new candidate enters the race as of 4/24/01.
            </value>
          </item>
        </value>
      </item>
    </value>
  </item>
  <item name="Author">
    <value>eal</value>
  </item>
  <item name="EMail">
    <value>eal@example.com</value>
  </item>
  <item name="Languages">
    <value>English</value>
  </item>
</record>
```

## Data Capture Example 2

The following sections show a hypothetical Yacht Information data capture form and the `datacapture.cfg` file that generates it.

### Example 2 Data Capture Form

The following is a hypothetical Yacht Information data capture form (browser interface). This form is included in the TeamSite Templating distribution and is available for use after you configure the example templating environment.

This example shows the usage of the `rowcontinue` and `colspan` elements used for placing multiple items on one line (supported only in TeamSite Templating Browser).

Teamsite Templating Data Capture Form - Microsoft Internet Explorer

File: Untitled Workarea: /default/main/cb1/WORKAREA/cw1

Save Save As Preview Generate Close

**internet/yacht**

- General Info**
  - Boat Manufacturer
  - Boat Model
  - Length
  - Rig
  - Hull Type
  - Hull Material
- Pricing**
  - Pricing-1**
    - Season
    - Time Periods**
      - Number of Cabins
      - Number of Staterooms
      - Included
      - Picture
      - Details

**Description:** Allows the entry of data relating to a sailing vessel and its seasonal charter prices.

**General Info**

Boat Manufacturer\*  Boat Model\*

Length\*  Rig\*

Hull Type\* ☒ Monohull ☐ Catamaran ☐ Trimaran

Hull Material\*

**Pricing-1**

Season\*

**Time Periods-1**

Time Period\*

Price\*

Number of Cabins\*

Number of Staterooms\*

Included ☐ Spinnaker ☐ Tri-sail ☐ Genoa ☐ Jib ☐ Storm Jib ☐ Dinghy

☐ Liferaft ☐ EPIRB

Picture

Details

Data Capture Form for Yacht datacapture.cff



## Example 2 datacapture.cfg File

The internet/yacht/datacapture.cfg file that generates this Yacht Information data capture form is as follows:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE datacapture SYSTEM "datacapture5.0.dtd">

<data-capture-requirements type="content" name="yacht">
  <!-- data-capture-requirements elements contain area elements -->
  <ruleset name="Vessel Information">

    <description>
      Allows the entry of data relating to a sailing vessel and its
      seasonal charter prices.
    </description>

    <container name="General Info" combination="and">
      <item name="Boat Manufacturer" rowcontinue="t">
        <database data-type="VARCHAR(40)" />
        <text required="t" maxlength="40" />
      </item>

      <item name="Boat Model" rowcontinue="f">
        <database data-type="VARCHAR(40)" />
        <text required="t" maxlength="40" />
      </item>

      <item name="Length" rowcontinue="t">
        <database data-type="SMALLINT" />
        <text required="t" maxlength="3" validation-regex="^[0-9]+?" />
      </item>

      <item name="Rig" rowcontinue="f">
        <database data-type="CHAR(6)" />
        <select required="t">
          <option selected="t" value="Sloop" label="Sloop"/>
          <option value="Ketch" label="Ketch"/>
          <option value="Cutter" label="Cutter"/>
        </select>
      </item>
    </container>
  </ruleset>
</data-capture-requirements>
```



```

<item name="Hull Type" colspan="2">
  <database data-type="CHAR(9)" />
  <radio required="t">
    <option selected="t" value="Monohull" label="Monohull"/>
    <option value="Catamaran" label="Catamaran"/>
    <option value="Trimaran" label="Trimaran"/>
  </radio>
</item>

<item name="Hull Material">
  <database data-type="VARCHAR(15)" />
  <select required="t">

    <!-- To use the example server-side
          inline callout, uncomment the
          next line for UNIX:

    <inline command="__IW_HOME__/iw-perl/bin/iwperl __IW_HOME__/examples/
    Templating/config/example_server_side_inline_callout.ipl" />

          or this line for Windows NT:

    <inline command="__IW_HOME__/iw-perl/bin/iwperl.exe __IW_HOME__/examples/
    Templating/config/example_server_side_inline_callout.ipl" />

          replacing "__IW_HOME__" with
          the location of your TeamSite
          installation. -->

    <option value="Fiberglass" label="Fiberglass"/>
    <option value="Wood" label="Wood"/>
    <option value="Steel" label="Steel"/>
    <option value="Aluminium" label="Aluminium"/>
    <option value="Ferrocement" label="Ferrocement"/>
    <option value="Other" label="Other"/>
  </select>
</item>
</container>

```



```
<item name="Pricing">
  <database deploy-column="f" />
  <replicant min="1" max="5">
    <item name="Season">
      <select required="t" multiple="t" size="5">
        <option selected="t" value="All Year" label="All Year"/>
        <option value="Winter" label="Winter"/>
        <option value="Spring" label="Spring"/>
        <option value="Summer" label="Summer"/>
        <option value="Fall" label="Fall"/>
      </select>
    </item>

    <item name="Time Periods">
      <replicant min="1" max="3">
        <item name="Time Period">
          <select required="t">
            <option value="Day" label="Day"/>
            <option value="Week" label="Week"/>
            <option value="Month" label="Month"/>
          </select>
        </item>

        <item name="Price">
          <description>Whole dollar amount</description>
          <text required="t" maxlength="10"
            validation-regex="^[0-9]+$" />
        </item>
      </replicant>
    </item>
  </replicant>
</item>

<item name="Number of Cabins">
  <database data-type="SMALLINT" />
  <select required="t">
    <option value="1" label="One"/>
    <option value="2" label="Two"/>
    <option value="3" label="Three"/>
    <option value="4" label="Four"/>
    <option value="5" label="Five"/>
    <option value="6" label="Six"/>
  </select>
</item>
```

```

<item name="Number of Staterooms">
  <database data-type="SMALLINT" />
  <select required="t">
    <option value="1" label="One"/>
    <option value="2" label="Two"/>
    <option value="3" label="Three"/>
    <option value="4" label="Four"/>
    <option value="5" label="Five"/>
    <option value="6" label="Six"/>
  </select>
</item>

<item name="Included">
  <database data-type="VARCHAR(32)" />
  <checkbox>
    <option value="Spinnaker" label="Spinnaker"/>
    <option value="Tri-sail" label="Tri-sail"/>
    <option value="Genoa" label="Genoa"/>
    <option value="Jib" label="Jib"/>
    <option value="Storm Jib" label="Storm Jib"/>
    <option value="Dinghy" label="Dinghy"/>
    <option value="Liferaft" label="Liferaft"/>
    <option value="EPIRB" label="EPIRB"/>
  </checkbox>
</item>

<item name="Picture">
  <database deploy-column="f" />
  <browser extns=".gif,.jpg"
    initial-dir="/templatedata/internet/yacht/images"/>
</item>

<item name="Details">
  <database deploy-column="f" />
  <textarea rows="5" cols="40"/>
</item>

</ruleset>
</data-capture-requirements>

```

## Data Capture Template DTD

The following code shows the `datacapture5.0.dtd` file that contains the syntax of the elements needed to create a `datacapture.cfg` file.

```
<!-- datacapture5.0.dtd -->

<!-- Start with some basic parameter entities. -->
<!ENTITY % items "container|item|itemref">
<!ENTITY % chooser-options "option|inline">

<!ELEMENT data-capture-requirements (symbol-table?,ruleset+)>
  <!ATTLIST data-capture-requirements
    name                CDATA                #IMPLIED
    type                (metadata|content|workflow) #REQUIRED
    dtd-system-identifier CDATA                #IMPLIED
  >
<!-- The 'dtd-system-identifier' attribute is a URI indicating the
DTD from whence a particular data-capture-requirements was
derived, if any.

In TeamSite Templating, the value of this attribute is used as
the system identifier of the document type declaration of a DCR
if and only if that DCR's type is "xml", as defined in
templating.cfg.
-->

<!ELEMENT symbol-table (symbol*) >

<!ELEMENT symbol          (%items;)? >
  <!ATTLIST symbol
    name                CDATA                #IMPLIED
    regex               CDATA                #IMPLIED
  >
```

```

<!ELEMENT ruleset (label?,description?,(%items;)*)>
  <!ATTLIST ruleset
    name          CDATA          #REQUIRED
  >

<!ELEMENT container (label?,description?,(%items;)* )>
  <!ATTLIST container
    name          CDATA          #REQUIRED
    hide-name     (t|f)         "f"
    combination   (and|or)      "and"
  >

<!ELEMENT itemref EMPTY >
  <!ATTLIST itemref
    name          CDATA          #REQUIRED
  >

<!ELEMENT item (label?,description?,database?(checkbox|radio|
  text|textarea|select|replicant|browser|
  readonly|hidden)+) >
  <!ATTLIST item
    name          CDATA          #REQUIRED
    rowcontinue   (t|f)         "f"
    colspan       CDATA          #IMPLIED
  >

<!ELEMENT label (#PCDATA) >
<!ELEMENT description (#PCDATA) >

<!ELEMENT text (allowed?,callout?,java-callout?,cgi-callout?,
  default?) >
  <!ATTLIST text
    required      (t|f)         "f"
    maxlength     CDATA          "0"
    size         CDATA          "0"
    validation-regex CDATA      #IMPLIED
  >
<!-- validation-regex is a Perl regex for validating this element -->

```



```
<!ELEMENT textarea      (allowed?,callout?,java-callout?,cgi-callout?,
                        default?) >
  <!ATTLIST textarea
    required      (t|f)          "f"
    rows          CDATA          "0"
    cols          CDATA          "0"
    wrap          (off|virtual|physical) "off"
    validation-regex CDATA        #IMPLIED
    rtf           (t|f)          "f"
    line-break    (P | BR)       #IMPLIED
    external-editor CDATA        " "
    external-editor-config CDATA  #IMPLIED
  >
<!-- validation-regex is a Perl regex for validating this element -->

<!ELEMENT browser      (allowed?,callout?,java-callout?,cgi-callout?) >
  <!ATTLIST browser
    required      (t|f)          "f"
    maxlength     CDATA          "0"
    size          CDATA          "0"
    initial-dir   CDATA          #IMPLIED
    ceiling-dir   CDATA          #IMPLIED
    extns         CDATA          #IMPLIED
  >

<!ELEMENT readonly     (allowed?,callout?,java-callout?,cgi-callout?) >

<!ELEMENT hidden       (allowed?,callout?,java-callout?,cgi-callout?) >
  <!ATTLIST hidden
    required      (t|f)          "f"
  >

<!ELEMENT checkbox     (allowed?,callout?,java-callout?,cgi-callout?,
                        (%chooser-options;)+) >
  <!ATTLIST checkbox
    required      (t|f)          "f"
    delimiter     CDATA          ", "
  >
```

```

<!ELEMENT radio      (allowed?,callout?,java-callout?,cgi-callout?,
                      (%chooser-options;)+) >
  <!ATTLIST radio
    required      (t|f)          "f"
  >
<!ELEMENT select     (allowed?,callout?,java-callout?,cgi-callout?,
                      (%chooser-options;)+) >
  <!ATTLIST select
    required      (t|f)          "f"
    size          CDATA          "0"
    multiple      (t|f)          "f"
    delimiter     CDATA          ", "
    width         CDATA          #IMPLIED
  >
<!-- The delimiter attribute is for multiple=t only -->

<!ELEMENT option     EMPTY >
  <!ATTLIST option
    selected      (t|f)          "f"
    value         CDATA          #IMPLIED
    label         CDATA          #REQUIRED
  >

<!ELEMENT replicant  (allowed?, (%items;)*)>
  <!ATTLIST replicant
    min           CDATA          "0"
    max           CDATA          "1"
    default       CDATA          "1"
    combination   (and|or)       "and"
    hide-name     (t|f)          "t"
  >

<!ELEMENT allowed   (cred|and|or|not)>

<!ELEMENT cred      EMPTY>
  <!ATTLIST cred
    role          CDATA          #IMPLIED
    user          CDATA          #IMPLIED
  >

```



```
<!ELEMENT and (cred|and|or|not)+>
```

```
<!ELEMENT or (cred|and|or|not)+>
```

```
<!ELEMENT not (cred|and|or|not)>
```

```
<!ELEMENT default (#PCDATA)>
```

```
<!-- The callout element is deprecated in favor of java-callout and  
cgi-callout -->
```

```
<!ELEMENT callout (param*) >  
  <!ATTLIST callout  
    type (java-class) #REQUIRED  
    label CDATA #REQUIRED  
    location CDATA #REQUIRED  
    class CDATA #REQUIRED  
    url CDATA #IMPLIED  
    window-features CDATA #IMPLIED  
  >
```

```
<!--The form of this element is exactly the same as that for the callout  
element in datacapture.4.0.dtd. -->
```

```
<!ELEMENT cgi-callout EMPTY>  
  <!ATTLIST callout  
    url CDATA #REQUIRED  
    label CDATA #REQUIRED  
    window-features CDATA #IMPLIED  
  >
```

```
<!--The form of this element is exactly the same as that for the callout  
element in datacapture4.5.dtd. -->
```

```
<!ELEMENT java-callout (param*) >  
  <!ATTLIST callout  
    label CDATA #REQUIRED  
    location CDATA #REQUIRED  
    class CDATA #REQUIRED  
  >
```



```

<!ELEMENT param          EMPTY >
  <!ATTLIST param
    name          CDATA          #REQUIRED
    value         CDATA          #REQUIRED
  >

<!ELEMENT database        EMPTY >
  <!ATTLIST database
    deploy-column  (t|f)         "t"
    searchable     (t|f)         "t"
    data-type      CDATA          "VARCHAR(255)"
    data-format    CDATA          #IMPLIED
  >

<!-- An 'inline' element should have a 'command' attribute. e.g.:
      <inline command="/bin/cat /tmp/a /tmp/b"/>

```

The callout program should return a well-formed XML document. The document's outermost element should be a "substitution" element. It should contain any XML that is valid according to this DTD.

That "substitution" element's contents will replace the "inline" element in the `datacapture.cfg` file.

So, if this DCT snippet:

```

<select>
<inline command="blah" />
</select>

```

runs the "blah" callout program, and that program returns this text:

```

<substitution>
<option label="ABC" />
<option label="123" />
<option label="Jackson 5" />
</substitution>

```



then the DCT snippet will, after callout execution and inline substitution, look like:

```
<select>
<option label="ABC" />
<option label="123" />
<option label="Jackson 5" />
</select>
-->
```

```
<!ELEMENT inline      EMPTY >
  <!ATTLIST inline
    command          CDATA          #REQUIRED
  >
```

# Setting Up Presentation Templates

---

## Creating Presentation Templates

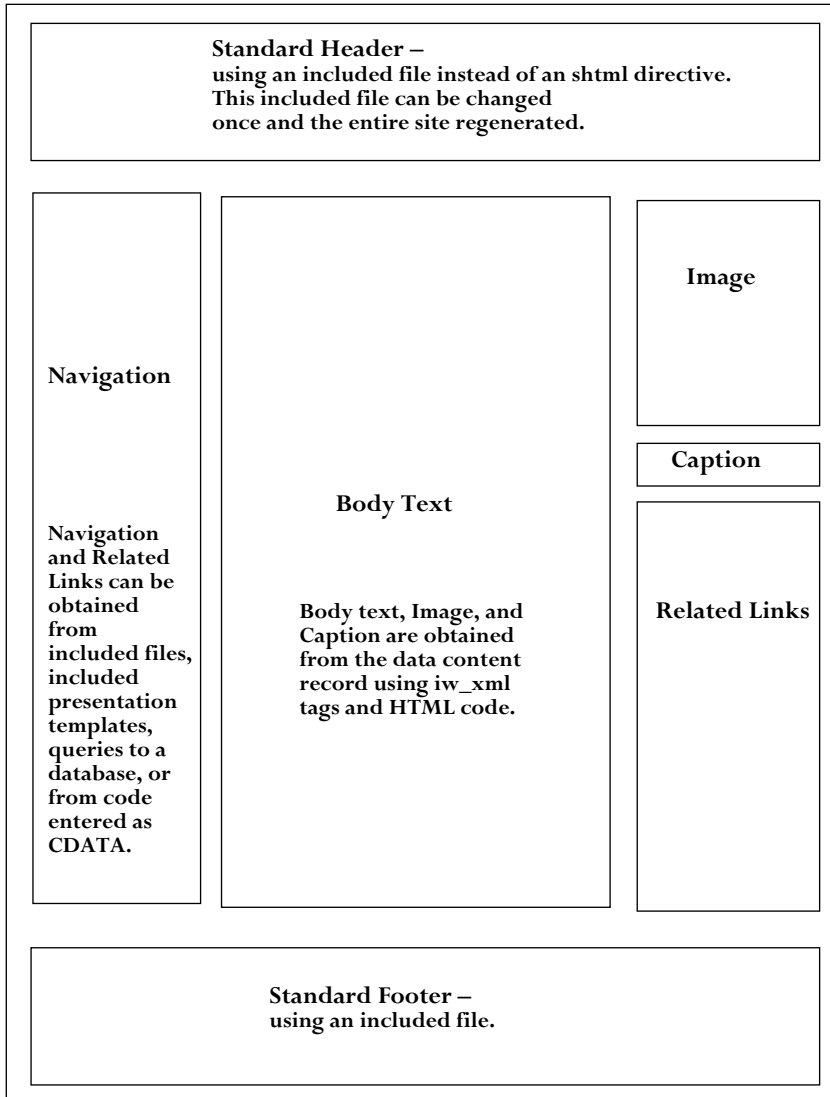
Presentation templates are designed to display data. The data may be obtained from the following sources:

- Data content records
- Queries to relational databases
- Perl-generated output
- Included files
- Included presentation components

You can combine data content records with presentation templates to generate output files. You can also create output files using relational database queries and output generated through the Perl API. TeamSite Templating can generate any text content, including HTML, XML, or any application server code. Using TeamSite Templating, you can precompile elements or a dynamic page, maintain dynamic content as application server code, eliminate the need for sever-side includes, and output an .asp or .jsp file that can be served dynamically at runtime in the production environment. At a minimum, TeamSite Templating can precompile flat HTML files that can sit as static files to provide maximum performance.

Presentation templates are written in XML and may contain custom Interwoven XML tags, HTML, and Perl.

The following diagram shows an example of how an output page may be generated when data is obtained from a data content record.



*Generating a Web Page with TeamSite Templating*

Presentation templates allow you to:

- Generate output files.
- Use built-in tags to fetch elements from XML data content records, loop on lists, do SQL queries, perform conditional logic, and so on.
- Create custom XML tags that encapsulate arbitrary presentation logic. Non-programmers can use custom high-level visual building blocks without writing any code.
- Create custom libraries and invoke them from within the `<iw_perl>` tag. Lower-level visual building blocks can be accessed by programmers directly from a template.
- Intermix XML and Perl to generate any output format (such as `html`, `asp`, and `jsp`). Presentation information does not need to be hard-coded into the template.
- Make common code components reusable across templates.
- Create generic components (component templates) that display differently based on the parameters they are given by their enclosing template.
- Eliminate page compilation costs on the production web server, thus increasing scalability of your web site.
- Use component templates. The component template may have key, value parameters passed to it by the enclosing template. For example, a component template may include an SQL query whose body depends on parameters from the enclosing template. Component templates do not take a data content record.

To write a presentation template, you must know some basic XML. Specifically, an understanding of the following XML topics is useful:

- CDATA
- “Well-formed” documents
- Entities
  - `&gt;`;
  - `&lt;`;
  - `&amp;`;
  - `&apos;`;
  - `&quot;`;

A useful reference is <http://www.xml.com/axml/testaxml.htm>.



Interwoven XML tags are an important part of writing presentation templates. The following is an overview of the existing tags:

<code>&lt;iw_xml&gt;</code>	Base class for presentation template XML elements.
<code>&lt;iw_pt&gt;</code>	Specifies that the document is a presentation template, and names it.
<code>&lt;iw_value&gt;</code>	Inserts the value of a Perl expression or data content record item.
<code>&lt;iw_load_dcr&gt;</code>	Used for loading data content records or arbitrary XML files dynamically.
<code>&lt;iw_if&gt;</code>	Provides an expression that is evaluated as being either true or false to determine whether the <code>&lt;iw_then&gt;</code> or <code>&lt;iw_else&gt;</code> statement will be used.
<code>&lt;iw_then&gt;</code>	Provides contents to be included if the <code>&lt;iw_if&gt;</code> tag's expression is true.
<code>&lt;iw_else&gt;</code>	Provides contents to be included if the <code>&lt;iw_if&gt;</code> tag's expression is false.
<code>&lt;iw_ifcase&gt;</code>	Provides for conditional inclusion of contents.
<code>&lt;iw_case&gt;</code>	Used with <code>&lt;iw_ifcase&gt;</code> for conditional inclusion of contents.
<code>&lt;iw_perl&gt;</code>	Executes arbitrary Perl code and provides an API for generating input and using data content records
<code>&lt;iw_ostream&gt;</code>	Used to change the default output stream (for multiple output).
<code>&lt;iw_iterate&gt;</code>	Iterates through a data content record or Perl list.
<code>&lt;iw-cscript&gt;</code>	Inserts the output of the Windows Script Host engine into a generated page.
<code>&lt;iw_sql_open&gt;</code>	Opens a database connection.
<code>&lt;iw_sql_iterate&gt;</code>	Iterates SQL result sets.
<code>&lt;iw_sql_query&gt;</code>	Queries a database.
<code>&lt;iw_system&gt;</code>	Uses output from an external command.
<code>&lt;iw_next&gt;</code>	Skips to the next iteration of a (possibly labeled) loop.
<code>&lt;iw_last&gt;</code>	Skips to the last iteration of a (possibly labeled) loop.
<code>&lt;iw_include&gt;</code>	Inserts a file or the result of compiling a template component in the generated HTML.
<code>&lt;iw_repeat&gt;</code>	Allows you to repeat content a given number of times.

Consider the following guidelines when creating a presentation template:

- When writing presentation templates that obtain information from data content records, refer to the data capture template that the data content records are based on. Make sure that the names of the fields are consistent and that you use `<iw_iterate>` tags in the presentation template if there are `replicant` tags in the data capture template (see Chapter 3, “Setting Up Data Capture Templates”).
- Presentation templates must be well-formed XML. Any HTML contained within a presentation template outside of a CDATA directive must be well-formed in accordance with XML rules.
- The `<iw_value>` tag, unlike all other tags, is also interpreted within CDATA sections. If you need to enclose a large body of text (e.g., HTML) with CDATA, you still have access to data values within this region.

## Using a Presentation Template—An Example

This section provides an overview to show the use of a presentation template. The section includes an example data content record, a presentation template, and a component template.

The presentation template shows how to use tags to call a component template, include a file, obtain data from a data content record, and iterate through all values of a field in a data content record.

The Press Release presentation template is shown on page 90. In addition to using HTML, it uses many of the `iw_xml` tags. This example is provided as `templatedata/internet/pr/presentation/nested_component_example.tpl` in your TeamSite Templating installation. This presentation template calls the `simple.tpl` component template (page 94) and accesses a data content record (page 88) to obtain values. The generated press release is shown (page 95).



The data content record that contains the data for the Press Release presentation template follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE record SYSTEM "dcr5.0.dtd">
<record name="pressrelease" type="content">
  <item name="Publish Date">
    <value>2001-04-19</value>
  </item>
  <item name="Headline">
    <value>Interwoven Announces TeamSite 5.0</value>
  </item>
  <item name="Secondary Headline">
    <value>April 19, 2001</value>
  </item>
  <item name="Introductory Paragraph">
    <value>Interwoven, Inc., the leading provider of enterprise-class
    content management today announced the 5.0 release of TeamSite,
    TeamSite Templating, and OpenDeploy software, along with the
    introduction of two new products -- Interwoven OpenChannel and
    MetaTagger 2.1 software.</value>
  </item>
  <item name="Story">
    <value>
      <item name="Subheading">
        <value>Content Infrastructure</value>
      </item>
      <item name="Section Paragraphs">
        <value>
          <item name="Paragraphs">
            <value>Interwoven also unveiled its new Content
            Infrastructure direction. This broad new product
            functionality addresses the customer need for Content
            Aggregation, Content Collaboration, Content Management,
            Content Intelligence and Content Distribution, to
            provide an underpinning for multiple eBusiness
            initiatives and applications.</value>
          </item>
        </value>
      </item>
    </value>
  </item>
</record>
```



```
<item name="Author">
  <value>eal</value>
</item>
<item name="EMail">
  <value>sales@interwoven.com</value>
</item>
<item name="Language">
  <value>English</value>
</item>
</record>
```



The presentation template for the press release follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<iw_pt name="Press Release 3"><![CDATA[
<HTML>
<!-- Begin CDATA Tag -->
<!-- HTML stuff is enclosed in CDATA tag -->

<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8">
</HEAD>
<BODY bgcolor="#FFFFFF" link="#0033CC" vlink="#0033CC" alink="0000FF"
TEXT=#000000 BACKGROUND="/templatedata/internet/pr/images/pixel.gif">
    <TABLE WIDTH="720" VALIGN="top" CELLPADDING="0" CELLSPACING="0"
        BORDER="0">
        <TR><TD WIDTH="720">

]]> <!-- End of CDATA Tag -->
<!-- nested component -->
    <iw_include pt='/templatedata/components/simple.tpl'>
        <![CDATA[
            $iw_param{Headline} = iwpt_dcr_value('dcr.Headline',
        ]]>
    </iw_include>

<![CDATA[ </TD></TR><TR><TD><TABLE WIDTH="720"><TR valign="top">
    <TD valign="top"> ]]>

        <iw_include file='templatedata/internet/pr/iwprnavbar.html' />

<![CDATA[ <!-- Begin CDATA Tag -->
    </TD><TD VALIGN="top" WIDTH="510">
]]> <!-- End of CDATA Tag -->

<!-- Begin content area -->

<!-- Headline -->
<P></P>
<br></br>

<!-- Secondary Headline -->
<h3> <iw_value name='dcr.Secondary Headline' /> </h3>
```

Using <iw\_pt> to open and name the presentation template; beginning CDATA

Using the <iw\_include> tag to call the simple.tpl component template

Passing "Headline" value as a parameter to component template

Using the <iw\_include> tag to include an HTML file

```

<!-- Date -->
<P> <B>SUNNYVALE, Calif., <iw_value name='dcr.Date' />:</B></P>

<!-- Introductory Paragraph -->
<P><iw_value name='dcr.Introductory Paragraph' /> </P>

<!-- Story -->
<iw_iterate var='story' list='dcr.Story'>

    <!-- Subheading -->
    <iw_value name='story.Subheading' />

        <!-- Paragraphs -->
        <iw_iterate var='para_value' list='story.Section Paragraphs'>
            <p><iw_value name='para_value.Paragraphs' /></p>
        </iw_iterate>
    </iw_iterate>

<!-- Insert 'aboutIW.html' file -->
<p>
<iw_include file='templatedata/internet/pr/aboutIW.html' />
</p>

<p>For more information on the company and
its software solutions, visit the Interwoven Web site at
<a href="http://www.interwoven.com">www.interwoven.com</a>
or e-mail <a href="mailto:{iw_value name='dcr.EMail'}">
    <iw_value name='dcr.EMail' /></a>
</p>

<!-- HTML stuff is enclosed in CDATA tag -->
<![CDATA[ <!-- Begin CDATA tag -->
<p>
<TABLE WIDTH=520 BORDER=0 CELLSPACING=10 CELLPADDING=0>
<TR>
    <TD COLSPAN=2 BGCOLOR=#999999>
        <IMG SRC="/templatedata/internet/pr/images/pixel.gif">
    </TD>
</TR>

```

Obtaining a value from a data content record using the <iw\_value> tag

Nesting of <iw\_iterate> tags

Using <iw\_iterate> and <iw\_value> to obtain multiple paragraph values

Opening CDATA containing HTML

[illegible]

```
<TR>
  <TD COLSPAN=2 ALIGN=CENTER>    <BR>
    <FONT COLOR=#666666>    <P CLASS="copyright">
      <A HREF="/copyright.html">    Copyright &copy; 2000</A>
      Interwoven, Inc. All rights reserved. </P> </FONT>
    </TD>
  </TR>
</TABLE>
</TD>
</TR>
</TABLE>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
]]>    <!-- End CDATA tag -->
</iw_pt>
```

Closing CDATA containing HTML

Closing the presentation template



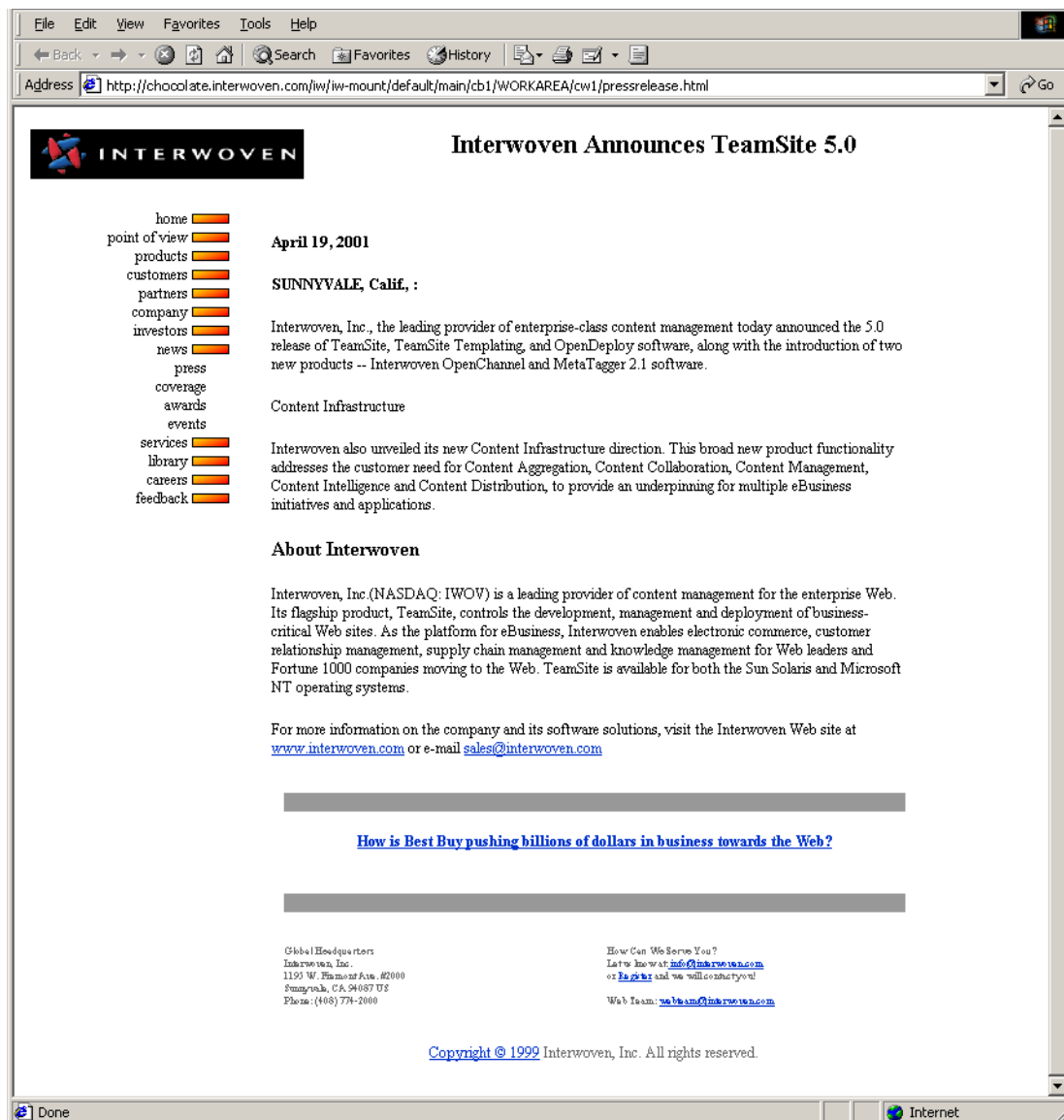
The `simple.tpl` component template is called from the main presentation template. It prints the headline it obtains from the calling presentation template. The contents of the `simple.tpl` component template is:

```
<?xml version="1.0" standalone="yes"?>
<iw_pt name="Banner Component PT">
<!-- This is a component PT that can be used inside another PTs -->
<!-- It prints the title that it got from the container PT -->
<TABLE width="720" align="center">
  <TR align="center">
    <TD align="left">
      <IMG SRC="/templatedata/internet/pr/images/iw-logo-small.gif"
        WIDTH="220" HEIGHT="40" BORDER="0"/>
    </TD>
    <TD align="center">
      <H1><iw_value name='$iw_arg{Headline}'/></H1>
    </TD>
  </TR>
</TABLE>
</iw_pt>
```

Tag that opens  
and names the  
component template

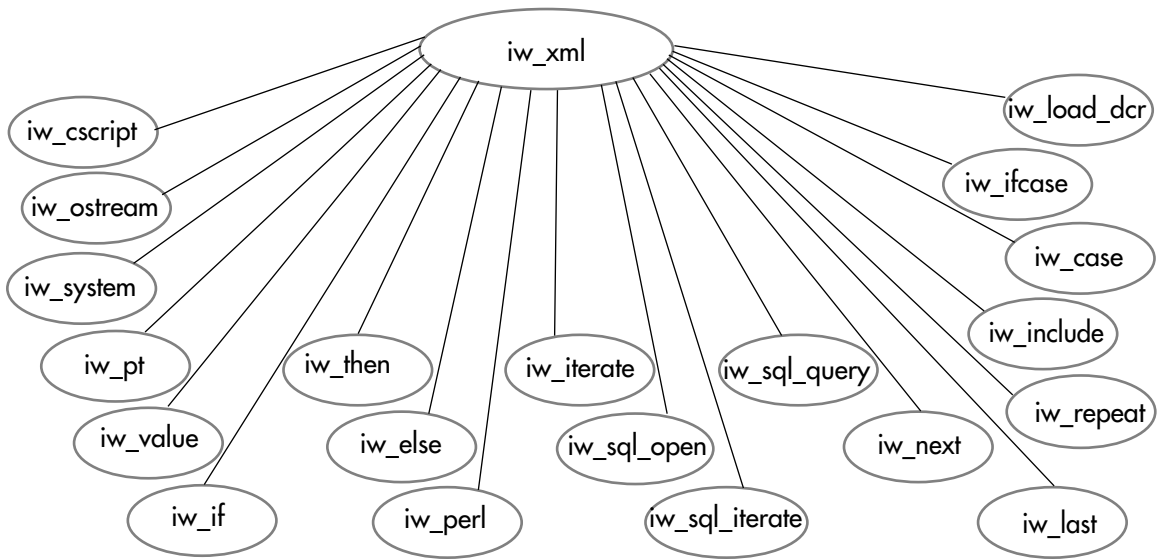
Tag that obtains the  
headline from the  
calling template

The press release generated from this presentation template, component template, and data content record would be as follows:



## Interwoven XML Tags

A number of custom Interwoven XML tags are supplied with TeamSite Templating. All of these tags are derived from a base class, `iw_xml`, as shown below.



Descriptions and examples of the `<iw_xml>` tags are provided online in HTML format within your installed version of TeamSite Templating. Go to <http://server/iw/help/tst/pt> for an index.

Descriptions of the tags in HTML format are also available at <http://support.interwoven.com/library/devel/tst/pt>. This site may also contain additional examples and other information you may find useful.

Typical man pages are available online for each tag. If `iw-home/iw-perl/bin` is in your path statement, you can access these man pages by issuing the command `perldoc TeamSite::PT::tag_name`.



# Mapping Users, Templates, and Content Records

---

This chapter describes how the `templating.cfg` file maps the interaction between content contributors, data capture templates, presentation templates, and data content records. Sections in this chapter discuss the following:

- An overview of `templating.cfg`.
- An example of a `templating.cfg` file.
- The `templating.cfg` DTD.

## templating.cfg Overview

The `templating.cfg` file is an XML file that resides outside of the TeamSite file system in `iw-home/local/config`. Each TeamSite Templating installation must have exactly one such file. By configuring `templating.cfg`, you can control:

- Which data categories and types TeamSite Templating can use.
- Which presentation templates can generate HTML files on which branches or directories.
- Which presentation templates can be used with a specific data type.
- Which users or roles are allowed to create or edit data content records for a specific data type.
- The location of the presentation template used for previewing generated HTML files.

The following sections describe how to perform these configurations.



## Example templating.cfg File

TeamSite Templating ships with the following sample `templating.cfg` file:

`iw-home/examples/Templating/config/templating.cfg.example`

This is the `templating.cfg` file that configures the example templating environment described in Chapter 2, “Initial Configuration.” The file configures TeamSite Templating to recognize and use the following data categories/types:

```
custom-dtd-example/PressRelease
intranet/deptinfo
intranet/weather
internet/careers
internet/auction
internet/pr
internet/book
internet/medical
internet/yacht
```

The following section shows a subset of this file with sections for the `PressRelease`, `deptinfo`, `weather`, `careers`, and `auction` data types.

See the diagram key on page 101 for details about items referenced in the `templating.cfg` file.

Note: The `templating.cfg` file must use either a UTF-8 or ISO-8859-1 encoding declaration. With any other encoding, **Preview** and **Generate** operations fail with a “malformed templating.cfg” error.

```

<?xml version="1.0" encoding="UTF-8"? standalone="no"?>
<!DOCTYPE templating SYSTEM "templating5.0.dtd">

<templating>
  <category name="custom-dtd-examples">
    <locations>
      <branch vpath-regex=".*" />
    </locations>
    <data-type name="PressRelease" dcr-type="xml"/>
    <presentation>
      <template name="PressRelease.tpl" extension="html">
        <locations>
          <branch vpath-regex=".*" preview-dir="/">
            <directory dir-regex=".*" />
          </branch>
        </locations>
      </template>
    </presentation>
  </category>
  <category name="intranet">
    <locations>
      <branch vpath-regex=".*" />
    </locations>
    <data-type name="deptInfo" dcr-type="iwov">
      <presentation>
        <template name="deptInfo.tpl" extension="html">
          <locations>
            <branch vpath-regex=".*" preview-dir="/">
              <directory dir-regex=".*" />
            </branch>
          </locations>
        </template>
      </presentation>
    </data-type>
  </category>
</templating>

```

Diagram illustrating the structure of the templating.cfg file with annotations:

- Templating Element <sup>1</sup> (points to <templating>)
- Data Category Section <sup>2</sup> (points to <category name="custom-dtd-examples">)
- Data Type Section <sup>3</sup> (points to <data-type name="PressRelease" dcr-type="xml"/>)
- Presentation Template Section <sup>4</sup> (points to <template name="PressRelease.tpl" extension="html">)
- Data Category Section <sup>2</sup> (points to <category name="intranet">)
- Data Type Section <sup>3</sup> (points to <data-type name="deptInfo" dcr-type="iwov">)
- Presentation Template Section <sup>4</sup> (points to <template name="deptInfo.tpl" extension="html">)
- Template to Data Type Mapping <sup>5</sup> (points to <branch vpath-regex=".\*" preview-dir="/"> inside the deptInfo template)
- Template to Generated File Mapping <sup>6</sup> (points to <directory dir-regex=".\*" /> inside the deptInfo template)
- Generated HTML File Location <sup>7</sup> (points to the <branch vpath-regex=".\*" preview-dir="/"> inside the deptInfo template)



```
<data-type name="weather" dcr-type="iwov">
  <presentation>
    <template name="weather.tpl" extension="html">
      <locations>
        <branch vpath-regex=".*" preview-dir="/">
          <directory dir-regex=".*" />
        </branch>
      </locations>
    </template>
  </presentation>
</data-type>
</category>
<category name="internet">
  <locations>
    <branch vpath-regex=".*" />
  </locations>
  <data-type name="careers" dcr-type="iwov">
    <presentation>
      <template name="jobDesc.tpl" extension="html">
        <locations>
          <branch vpath-regex=".*" preview-dir="/">
            <directory dir-regex=".*" />
          </branch>
        </locations>
      </template>
    </presentation>
  </data-type>
  <data-type name="auction" dcr-type="iwov">
    <presentation>
      <template name="auction.tpl" extension="html">
        <locations>
          <branch vpath-regex=".*" preview-dir="/">
            <directory dir-regex=".*" />
          </branch>
        </locations>
      </template>
    </presentation>
  </data-type>
</category>
</templating>
```

## Diagram Key

1. **Templating Element:** The `<templating>` element marks the beginning of the `templating.cfg` file's configuration information and identifies the file as a `templating.cfg` file.
2. **Data Category Section:** The `<category>` element contains information specific to a data category (intranet in this example) and makes the data category available for use by TeamSite Templating. The `<category>` element contains one or more `<data-type>` elements. A data category must have its own `<category>` element in `templating.cfg` for TeamSite Templating to recognize and use the data category. Even if a data category is located correctly in the directory structure described on page 23, it will not be recognized by TeamSite Templating unless it is named in a `<category>` element as shown here. The `<category>` element's name attribute is required. You can use the `<locations>` element within a `<category>` element to show the branches in which that category will be available. This example shows that `intranet` and `custom-dtd-examples` categories will be available in all branches.
3. **Data Type Section:** The `<data-type>` element contains information specific to a data type and makes the data type available for use by TeamSite Templating. A data type must have its own `<data-type>` element in `templating.cfg` for TeamSite Templating to recognize and use the data type. Even if a data type is located correctly in the directory structure described on page 23, it will not be recognized by TeamSite Templating unless it is named in a `<data-type>` element as shown here. The attributes for `<data-type>` are `name`, `items-per-page`, and `dcr-type`. The `<data-type>` element's name attribute is required. The `items-per-page` attribute is used to specify the number of items on each page in a data capture form. The `items-per-page` attribute applies only to top-level items; therefore, a replicant with many items still displays on one page and this attribute has no impact. The `dcr-type` specifies what kind of DCR to write out. The values are `xml` and `iwov`; the default is `iwov`. If the value of `dcr-type` is `xml`:
  - The data capture template for that data type needs to have been generated using `iwtdtd2sym`.
  - The data content records for that data type will be XML documents written according to the DTD that the data capture template was derived from.

The `<data-type>` element can contain the following subelements:

- `<locations>`: Shows the branches in which that data type will be available.
- `<presentation>`: See Item 4 below.



- **<allowed>**: Lets you set an ACL to specify which users can or cannot use a specific data type. If **<allowed>** is not set, any user can use the data type (see page 60 for additional examples). The **<allowed>** element can have any of the following subelements:

- <cred>**: Lets you name a user or role in the ACL (for example, `user="joe"` or `role="master"`).
- <and>**: Logical and statement for grouping ACL credentials.
- <or>**: Logical or statement for grouping ACL credentials.
- <not>**: Logical not statement for negating ACL credentials. For example, the following allows all users except joe to use the current instance:  

```
<allowed>  
  <not>  
    <cred user="joe">  
  </cred>  
  </not>  
</allowed>
```

4. **Presentation Template Section**: The **<presentation>** element marks the beginning of the section that contains subelements for presentation template mapping. See Items 5, 6, and 7 below.
5. **Template to Data Type Mapping**: The **<template>** element marks the beginning of the section that maps a presentation template to a data type. It specifies which presentation templates are available for use with the data type named in the **<data-type>** element. In the example shown here, the `deptInfo.tpl` template can be used to display data content records for the `deptinfo` data type. The **<template>** element can contain the following attributes:
  - **extension**: Specifies the extension that will be used on any files this template generates. This attribute is required.
  - **fullpage**: Specifies that the generated HTML file is a full HTML page. This attribute is optional.
  - **name**: Specifies the presentation template's file name in the *workarea\_name/templatedata/data\_category/data\_type/presentation* directory. This attribute is required.

6. **Template to Generated File Mapping:** The `<branch>` element uses extended regex syntax to specify on which branches a presentation template can generate a file. The `<branch>` element can have the following attributes:
- `vpath-regex`: Specifies on which branches files can be generated using this presentation template. The example shown here ("`.*`") specifies that all branches can have files generated by the `deptInfo.tpl` presentation template.
  - `preview-dir`: Specifies the directory (in an area of a branch) in which generated files will be previewed in when you preview a data content record (via the TeamSite Templating **Preview** button).
7. **Generated HTML File Locations:** The `<directory>` element uses regex syntax to specify where generated HTML files based on this presentation template may be saved. This example specifies that generated HTML files based on `jobDesc.tpl` will reside in the current directory (`.*`). The `<directory dir-regex="..." />` regular expression matches a directory relative to the user's workarea. Because the string that is matched against the regex does not begin with a slash, it is possible for the string to be empty (that is, when the directory in question is the top of the workarea, then an empty string will be matched against the regex).

## Setting Previewing Path Variables

The following example describes what happens when a user previews a generated HTML file in TeamSite Templating.

If the file is specified with an absolute path (for example, `href=/main/images/pixel.gif`), the browser searches the absolute path.

The way to configure TeamSite Templating so that the correct directory is searched is to set `preview-dir` in the `templating.cfg` file to point to the directory containing the file. For example, set the `preview-dir` variable to `/images` if `pixel.gif` resides in `/images`. Then `pixel.gif` will be found and displayed during the preview.

To summarize the preview results:

- If the line `href=pixel.gif` appears in the presentation template and the directory containing `pixel.gif` is named with the `preview-dir` variable in `templating.cfg`, `pixel.gif` will be included in the preview.



- If the line `href=absolute_path_name/pixel.gif` appears in the presentation template, the file `pixel.gif` will be included in the preview.

The *preview-dir* variable (in the `templating.cfg` file) associated with each presentation template defines the directory where the preview file will virtually exist during preview time. A preview creates a temporary file. When a browser is opened and directed to the preview file, the URL that the browser points to is the URL for the preview file in the directory defined in *preview-dir*. During the preview, a proxy remap occurs, remapping the directory specified in the *preview-dir* variable to the `templatedata/iw_preview` directory. In this way, a preview file can have a virtual location other than its true location. These temporary files are deleted by the previewing system.



## templating.cfg DTD

```

<!ELEMENT templating (category*) >

<!ELEMENT category (locations?,data-type*) >
    <!ATTLIST category
        name          CDATA          #REQUIRED
    >
<!ELEMENT data-type (locations?,allowed?,presentation?) >
    <!ATTLIST data-type
        name          CDATA          #REQUIRED
        items-per-page CDATA          #IMPLIED
        dcr-type      (iwov|xml)     "iwov"
    >

<!ELEMENT presentation (template*) >

<!ELEMENT template (locations) >
    <!ATTLIST template
        name          CDATA          #REQUIRED
        fullpage      t|f            "f"
        extension     CDATA          #REQUIRED
    >

<!ELEMENT locations (branch+) >

<!ELEMENT branch (directory*) >
    <!ATTLIST branch
        vpath-regex   CDATA          #REQUIRED
        preview-dir    CDATA          #IMPLIED
    >

<!-- 'branch' elements should only contain 'directory' elements
when they are within a 'template' element.
The 'preview-dir' attribute is required when the 'branch' element
is within a 'template' element. -->

```



```
<!ELEMENT directory EMPTY >
  <!ATTLIST directory
    dir-regex      CDATA      #REQUIRED
  >

<!-- This is the same stuff as datacapture5.0.dtd: -->

<!ELEMENT allowed (cred|and|or|not) >

<!ELEMENT cred EMPTY >
  <!ATTLIST cred
    role          CDATA      #IMPLIED
    user          CDATA      #IMPLIED
  >

<!ELEMENT and (cred|and|or|not)+ >

<!ELEMENT or (cred|and|or|not)+ >

<!ELEMENT not (cred|and|or|not) >
```

# Integrating Templating, DataDeploy, and Workflow

---

This chapter describes how to integrate TeamSite Templating with DataDeploy and TeamSite workflow. Integrating these components allows a content contributor to access a data capture template, create a data content record, and deploy the data content record's extended attributes to a database via a TeamSite workflow job. All of these activities take place as a single, integrated sequence of steps initiated and executed from the TeamSite GUI. The entire DataDeploy process runs as a TeamSite workflow job, so the content contributor does not need to start DataDeploy manually, or even be aware that DataDeploy is running.

**Note:** The configuration steps described in this chapter assume that TeamSite Templating is installed and configured as described in Chapter 2, “Initial Configuration.” DataDeploy must also be installed on your system.

Refer to the *TeamSite User's Guide* for information on using templating. Refer to the *TeamSite Administration Guide* for information on setting up TeamSite. Refer to the *DataDeploy Administration Guide* for information on setting up and using DataDeploy.

## Integration Overview

The following steps show the process to create, save, submit, and deploy a data content record when TeamSite Templating and DataDeploy are integrated.

1. In the TeamSite GUI, a content contributor requests a new data content record, chooses a data type, and enters data in the resulting data capture form.
2. The content contributor saves the data capture form.
3. In the TeamSite GUI, the content contributor selects a data content record and clicks **Submit**. Templating can be configured to automatically initiate a workflow process after a particular user action as a convenience to the end user. This can be done in `available_templates.cfg` (see “Editing `available_templates.cfg` to Initiate Workflows” on page 33).
4. DataDeploy is automatically signaled to perform the following functions:
  - Determine which data types are affected by the data content record change.
  - Read in all necessary database mapping information from DataDeploy configuration files.
  - Populate the database with elements of the data content record, based on the mapping file.
  - Write a log of all DataDeploy activity to the `dd-home/log` file.Refer to the *DataDeploy Administration Guide* for additional information.

## Integration Steps

The following sections describe the configuration steps you must perform on TeamSite Templating, TeamSite workflow, and DataDeploy to integrate them for your specific templating environment.

### TeamSite Templating

Install and set up TeamSite Templating as described in Chapter 2, “Initial Configuration,” which prepares TeamSite Templating for integration with DataDeploy and TeamSite workflow. You do not need to perform any additional tasks on TeamSite Templating to enable integration.

## DataDeploy

A DataDeploy configuration file must be created for each type of data content record that will be deployed. DataDeploy generates these configuration files automatically. However, the information is provided here for your information. For example, to use DataDeploy to deploy a data content record that is based on the data capture template `/templatedata/beverages/tea/datacapture.cfg`, a DataDeploy configuration file must be created for the data type `tea`. Likewise, to deploy a data content record based on `/templatedata/beverages/coffee/datacapture.cfg`, a DataDeploy configuration file must be created specifically for the data type `coffee`.

DataDeploy configuration files for TeamSite Templating use the following location and naming conventions:

`workarea_name/templatedata/data-category/data-type/data-type_dd.cfg`

For example:

`/workarea_name/templatedata/beverages/tea/tea_dd.cfg`

Or, in the case of the Press Release example shown in “Data Capture Example 1” on page 42:

`/workarea_name/templatedata/internet/pr/pr_dd.cfg`

Refer to the information in the *DataDeploy Administration Guide* for information on creating the DataDeploy configuration files and the database tables.

## TeamSite Workflow

This release of TeamSite Templating supports a preconfigured templating-specific workflow template, `author_submit_dcr.wft`. This file is installed by TeamSite Templating in `iw-home/local/config/wft/default`. It configures the **Author DCR Submit** workflow job displayed in the New job window when TeamSite Templating starts a workflow job. The files `author_submit_dcr-0.ipl` and `author_submit_dcr-0.ipl` were installed in `iw-home/local/bin` during the TeamSite Templating installation. Check `available_templates.cfg` to verify that the workflow is set up and to add additional workflows. See the *TeamSite Administration Guide* for an example of the TeamSite GUI’s New Job window.



## Appendix A

# Using Callouts

---

Three callout elements are defined in the `datacapture.dtd` file. The `<callout>` subelement is maintained for compatibility with previous versions of TeamSite Templating. The `<java-callout>` is designed for use with Java-based templates. The `<cgi-callout>` is designed for use with browser-based templates.

## The Java Callout

The `<java-callout>` subelement creates a button on the data capture form that can be programmed to call a Java program. An interface is provided that defines the `IWDataCaptureCallout` interface. You need to write a Java class that implements the interface. Java documentation (javadoc) that describes the API is available once you install TeamSite Templating. You can access this Javadoc through a browser at <http://TeamSite-server/iw/java-callout-api/overview-tree.html>. Source code and example classes can be accessed at `iw-home/local/config/java-callout-api`.

The Java callout shown in this section displays the content of a URL in a separate window. This example demonstrates some basic mechanics of a callout, such as passing parameters, getting the name of the field from whence the callout was called, and getting that field's value.

If your callout references a specific package in a jar format, put these jars in the same location as the callout. If a third party library is in a jar file, you should include your callout classes in the jar file (remembering to preserve the package structure of your classes). Remember that when referencing the location of a Java class in the `datacapture.cfg` file that it points to the top level where the package begins. Therefore, referencing the class `com.interwoven.dc100.api.classname` where the location in the data capture template is `/iw/java-stuff` means that the file `classname.class` is in `iw-home/httpd/webapps/iw/java-stuff/com/interwoven/dc100/api`.



An example of the section of the `datacapture.cfg` file and the Java source code file are shown here.

## The `datacapture.cfg` File

The following code should be added to the `datacapture.cfg` file to create the Java callout button:

```
<item name="ViewURL">
  <text size="10">
    <java-callout type='java-class' label='View URL'
      location='http://localhost/iw/java-callout-api/examples/'
      class='com.interwoven.dc100.api.ViewURLCallout'>
      <param name='URL' value='http://myServer/bazaar/baubles.html' />
      <param name='height' value='400' />
      <param name='width' value='300' />
    </java-callout>
  </text>
</item>
```

You must set the parameters `URL`, `height`, and `width`. The `height` and `width` parameters specify the callout window's dimensions; `URL` specifies the URL to display in this window.

## Java Source Code

The Java source code is shown in this section. In this example, the file was named `ViewURLCallout.java`. The Java source code must be compiled into a Java class file named `ViewURLCallout.class`. Copy the class file into its proper home. In this example, place the file `ViewURLCallout.class` into the directory `<iw-home>/httpd/iw/java-callout-api/examples/com/interwoven/dc100/api`. The class file is stored in the directory that corresponds to the `location` attribute of the `<java-callout>` element in your `datacapture.cfg` file.



```

////////////////////////////////////
// CLASS      ViewURLCallout
// VERSION    1.0.0
// DATE       March 2001
// COMPANY    Interwoven
// PURPOSE    Display a URL in a frame for a callout button.
//            Tested with TeamSite Templating 4.5.1
//            The parameters -- URL, height, width -- should be
//            defined in your datacapture.cfg file.
//
//            Note: To make the URL a bit more dynamic, you
//                  could use the value of the datacapture
//                  field as a parameter to a URL which uses
//                  it to generate a page (e.g., a Perl CGI
//                  script that looks up the value in a
//                  database and creates a product info page
//                  complete with images [see code below]).
//
////////////////////////////////////

package com.interwoven.dc100.api;

import java.awt.*;
import java.awt.event.*;
import javax.swing.JFrame;
import javax.swing.JEditorPane;
import javax.swing.JScrollPane;
import java.io.*;
import java.net.URL;
import java.util.Enumuration;

public class ViewURLCallout
    implements IWDDataCaptureCallout
{
    String      fieldName;           // Name of the DCT item
    String      fieldValue;         // Value of the DCT field
    Integer     width;              // width of callout frame
    Integer     height;             // height of callout frame
    String      URLstring;          // URL to display in callout
    JEditorPane editorPane;

```



```
public void      setParams( IWDataCaptureCalloutParams params )
{
    // Parameters defined in datacapture.cfg's <param> tags.
    width  = width.valueOf( params.getParameter("width") );
    height = height.valueOf( params.getParameter("height") );
    URLstring = params.getParameter("URL");
}

public void      start( IWDataCaptureRootNode root,
    IWDataCaptureObject object, JFrame DCframe )
{
    // Get name of datacapture field and its value.
    try{
        fieldName = object.getUniqueName();
        fieldValue = getFirstValue( (IWDataCaptureModifiableLeaf) object );
    }catch( IWDataCaptureException iwe ){

        JFrame frame = new JFrame();
        frame.getContentPane().setLayout(new BorderLayout() );
        // Create an editor pane.
        JEditorPane editorPane = createEditorPane();

        // Put the editor pane into a Scroll Pane.
        JScrollPane urlScrollPane = new JScrollPane(editorPane);
        urlScrollPane.setPreferredSize(new Dimension(width.intValue(),
            height.intValue() ));
        //frame.setTitle( "URL: " + URLstring + "    field name: " + fieldName +
        //                "    field value: " + fieldValue );
        frame.setTitle( URLstring );
        frame.getContentPane().add(urlScrollPane);
        frame.pack();
        frame.setVisible(true);

    return;
}
```

```

protected String getFirstValue( IWDDataCaptureModifiableLeaf leaf )
{
    // Return the first value of the field, if any.
    Enumeration values = leaf.getValues();
    if (values.hasMoreElements()) {
        String value = (String) values.nextElement();
        return value;
    }
    return "";
}

private JEditorPane createEditorPane() {
    JEditorPane editorPane = new JEditorPane();
    editorPane.setEditable(false);
    try {
        URL theURL = new URL( URLstring );

        // A more dynamic URL, using a query string.
        // URLstring = "http://myServer/scripts/productInfo.ipl";
        // URL theURL = new URL( URLstring + "?product=" + fieldValue );

        displayURL(theURL, editorPane);
    } catch (Exception e) {
        System.err.println("Couldn't create URL: " + URLstring);
    }

    return editorPane;
}

private void displayURL(URL url, JEditorPane editorPane) {
    try {
        editorPane.setPage(url);
    } catch (IOException e) {
        System.err.println("Attempted to read a bad URL: " + url);
    }
}
}

```

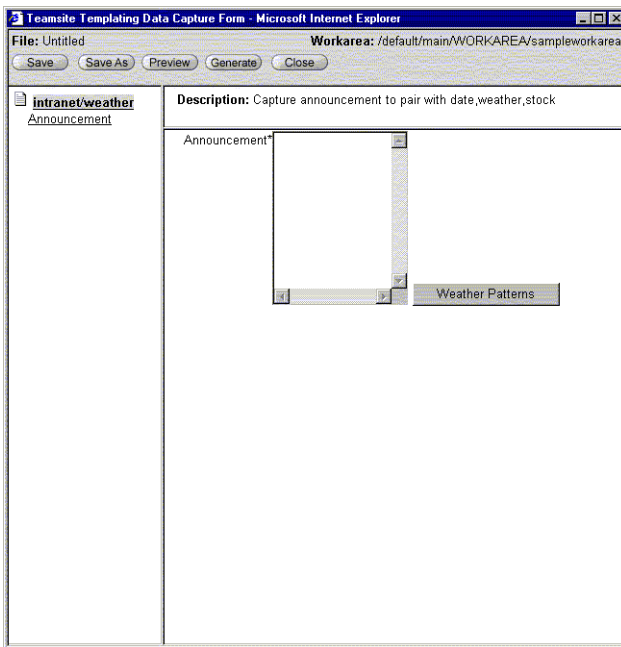
## The CGI Callout

This section describes an example of using the `<cgi-callout>` element. It shows an example of the data capture record that contains a callout button. It also shows the `datacapture.cfg` and `example_datacapture_callout.ipl` files that define and call the CGI.

The `<cgi-callout>` subelement can be used with the `<browser>`, `<checkbox>`, `<hidden>`, `<radio>`, `<readonly>`, `<text>`, `<textarea>`, and `<select>` elements. Refer to the Chapter 3, “Setting Up Data Capture Templates,” for information.

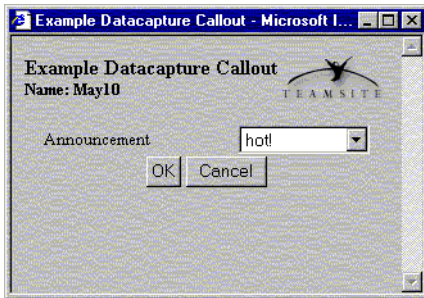
## The Data Capture Form

The following data capture form shows the CGI callout button, labeled **Weather Patterns**.



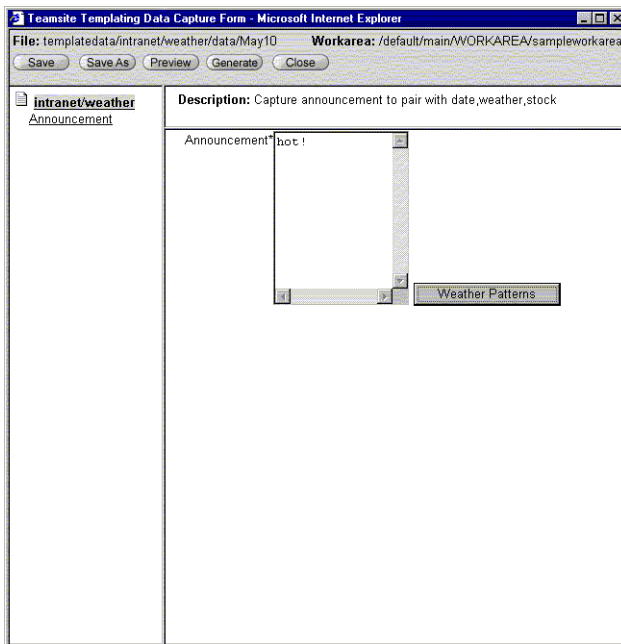
*Data Capture Form with a Callout for the Weather Patterns Item*

When a user clicks **Weather Patterns**, a dialog box displays so the user can select a value.



*Selecting a Value for Weather Patterns Using the Data Capture Callout*

In this example, the user selects “hot!” from the menu and clicks **OK**. This value is reflected in the data capture form.



*Data Capture Form with Announcement Field Reflecting the Value Selected using the Weather Patterns Button*



## The datacapture.cfg.example File

This section shows the `intranet/weather/datacapture.cfg.example` file that created the data capture form on page 116. The file contains the entries you need to use the callout CGI capability. To use this file to demonstrate the example, rename the file `datacapture.cfg`.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE datacapture SYSTEM "datacapture5.0.dtd">

<data-capture-requirements type="metadata">
  <!-- data-capture-requirements elements contain area elements -->
  <ruleset name="Interwoven Information Page">
    <description>
      Capture announcement to pair with date,weather,stock
    </description>

    <item name="Announcement">
      <textarea cols="15" rows="10" required="t">
        <cgi-callout url="/iw-bin/example_datacapture_callout.cgi/
options.txt" label="Weather Patterns" window-features="width=320,
height=200,resizable=yes,toolbar=no,scrollbars=yes"/>
      </textarea>
    </item>
  </ruleset>
</data-capture-requirements>
```

The following labels and value are included in the `options.txt` file described in the next section. These values are in the pull-down for the user selection in the **Weather Patterns** field.

```
hot!,hot!
sunny,sunny
cloudy,cloudy
partly cloudy,partly cloudy
drizzle,drizzle
showers,showers
rain,rain
sleet,sleet
snow,snow
```

## The example\_datacapture\_callout.ipl File

This section contains the `example_datacapture_callout.ipl` file. This file operates as a data capture callout CGI program to populate a data capture item with a selection from a dynamically generated list of options that are read from a flat file. This file does not need to be modified for the demonstration. You can rename and modify it as needed for customization.

The program reads a flat file of selection options and presents them to the user. The selected option is used to populate the data capture item that launched the callout. The data capture item must be `text` or `textarea` for this example.

The flat file is specified in the `PATH_INFO` part of the URL. The CGI is designed to look for the flat file (`options.txt`) in `iw-home/local/config`. The following URL specifies the flat file:

```
/iw-bin/iw_cgi_wrapper.cgi/example_datacapture_callout.ipl/options.txt
```

The format of each line in the flat file is:

```
value,label
```

An empty label results in a label identical to the value.

These following file contents would result in an HTML user interface containing the HTML shown below the file contents:

Flat file contents in the `value, label` format:

```
P,Paperback
H,Hardback
N,No back
XYZ,
```

HTML file:

```
<SELECT>
  <OPTION VALUE="P">Paperback</OPTION>
  <OPTION VALUE="H">Hardback</OPTION>
  <OPTION VALUE="N">No back</OPTION>
  <OPTION VALUE="XYZ">XYZ</OPTION>
</SELECT>
```



To ensure that any data transfer from the callout window to the data capture form is saved to the internal data structure, the Javascript in the callout window should make a call to `datacapture.refreshForm()`, a function located in the top level of the data capture window. For example, if the function is being called from the callout window that was opened by the data capture form (that is, not in a child window of the callout window), the call would be:

```
opener.top.datacapture.refreshForm()
```

This function takes no arguments and should be called before the callout window is dismissed. One way to do this is to put this call in the onclick handler of the **OK** button, as shown in this example. Another way is to put this call in the onUnload handler of the callout body so get called when the user dismisses the window. This call could be expensive so it should only be done sparingly.

### Data Capture Callout CGIs

These standard name-value pairs will be POSTed to the callout CGI. These name-value pairs are:

- `area_path`: The vpath to the current workarea. Unlike the custom menu item's `area_path`, this is a vpath rather than a file system path.
- `iw_callback_var`: The name of the form element that launched this callout.
- `iw_dcr_path`: A file system path to the data content record being edited.
- `iw_dcr_type`: The type of the data content record being edited.
- `iw_dcr_vpath`: A vpath to the data content record being edited.
- `iw_field_type`: The type of form element that launched this callout.
- `iw_form_name`: The name of the form that launched this callout. This is the full name relative to the callout window (that is, `window.opener.top.formframe.document.dcreditForm`).
- `iw_item_description`: The description of the data capture item that launched this callout.
- `iw_item_name`: The name of the data capture item that launched this callout.
- `iw_item_value`: The current value of the data capture item that launched this callout. This only applies for text and textarea items. Values of other types of form instances (check boxes, select menus, etc.) can be obtained through JavaScript.
- `iw_object_name`: The name of the data content record being edited.
- `session`: The TeamSite session string for the current session. This is the same as the custom menu item's `session`.



- `task_id`: The task ID for the current workflow task, if the data capture form was a cgitask of a workflow.
- `user_name`: The user name of the current TeamSite user. This is the same as the custom menu item's `user_name`.
- `user_role`: The role of the current TeamSite user. This is the same as the custom menu item's `user_role`.

**example\_datacapture\_callout.ipl**

```
use TeamSite::CGI_lite;
use TeamSite::Config;

$|=1;

my $cgi = TeamSite::CGI_lite->new();
$cgi->parse_data();

my $form_name           = $cgi->{'form'}{'iw_form_name'};
my $element_name        = $cgi->{'form'}{'iw_callback_var'};
my $user_name           = $cgi->{'form'}{'user_name'};
my $item_name            = $cgi->{'form'}{'iw_item_name'};
my $item_description     = $cgi->{'form'}{'iw_item_description'};
my $dcr_name             = $cgi->{'form'}{'iw_object_name'};

my $flat_file           = $ENV{'PATH_INFO'};
if ($flat_file =~ m!^\.*\/([\^\/]+)$!) {
    $flat_file = $1;
}

my @options              = parse_flat_file( make_full_filename( $flat_file ) );
if (! @options)
{
    error_message( 'No available selection options were found.' );
    exit 1;
}

print_ui( @options );

exit 0;

sub make_full_filename {
    my ($filename) = @_;
    my $iw_home = TeamSite::Config::iwgethome();
    my $full_filename = $iw_home . "/local/config/" . $filename;
    return $full_filename;
}
```

```

sub parse_flat_file {
    my ($filename) = @_ ;
    my @options;

    if ((-f $filename) && (open( FILE, $filename )))
    {
        while (<FILE>)
        {
            my $line = $_;
            if ($line =~ /^([^\,]*)\.(\.*)$/)
            {
                my %hash = ( 'value' => $1, 'label' => $2 );
                push( @options, \%hash );
            }
        }
    }

    close FILE;

    return @options;
}

sub print_ui {
    my (@options) = @_ ;

    print_header();2

    print <<"END";
<FORM NAME="callout_form">
<TABLE BORDER="0" CELLPADDING="0" ALIGN="CENTER">
<TR>
<TD VALIGN="TOP" WIDTH="150">
<FONT SIZE="-1">$item_name</FONT>
<BR>
<FONT SIZE="-1"><EM>$item_description</EM></FONT>
</TD>
<TD VALIGN="TOP">
<SELECT NAME="selection_list">
END

```



```
        foreach my $hash (@options)
        {
            my $value = $hash->{'value'};
            my $label = $hash->{'label'};
            if ($label eq '')
            {
                $label = $value;
            }
            print "<OPTION VALUE=\""$value\"">$label</OPTION>\n";
        }

print <<"END";
</SELECT>
</TD>
</TR>
</TABLE>
<CENTER>
<INPUT TYPE="BUTTON" VALUE="OK"      onClick="handle_selection()">
<INPUT TYPE="BUTTON" VALUE="Cancel"  onClick="self.close()">
</CENTER>
</FORM>
END

    print_footer();

    return;
}
sub error_message {
    my (@msgs) = @_;

    print_header();
    foreach my $message (@msgs)
    {
        print $message;
    }

    print <<"END";
```

```

<CENTER>
<FORM>
<INPUT TYPE="BUTTON" VALUE="OK" onClick="self.close()">
</FORM>
</CENTER>
END

    print_footer();
    return;
}

sub print_header {
    print<<"END";
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Example Datacapture Callout</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--

function set_datacapture_item_value( selectedValue )
{
    if ((window.opener == null)    ||
        (window.opener.closed))
    {
        return false;
    }

    var calloutForm = eval($form_name);
    if (!calloutForm)
    {
        return false;
    }
}

```



```
var calloutElementFound = false;
for ( i = 0 ; i < calloutForm.elements.length ; i++ )
{
    if (calloutForm.elements[i].name == '$element_name')
    {
        calloutForm.elements[i].value = selectedValue;
        calloutElementFound = true;
        break;
    }
}
if (!calloutElementFound)
{
    return false;
}

return true;
}

function handle_selection()
{
    if (callback())
    {
        if(opener.top.datacapture) {
            opener.top.datacapture.refreshForm();
        }
        self.close();
    }
    else
    {
        alert('Please make a selection.');
```

```

function callback()
{
    var optionsArray = document.callout_form.selection_list.options;
    for ( i = 0 ; i < optionsArray.length ; i++ )
    {
        if (optionsArray[i].selected)
        {
            if (!set_datacapture_item_value( optionsArray[i].value ))
            {
                alert('Fatal callout error. Did you close the datacapture
window?');
            }
            return true;
        }
    }
    // did not find a selected option!
    return false;
}

// -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#CoCoCo">
<TABLE BORDER=0 cellpadding=0 cellspacing=0 WIDTH=100%>
    <TR>
        <TD ALIGN="LEFT"><B>Example Datacapture Callout</B><BR>
        <FONT SIZE="-1"><B>Name: $dcr_name</B></FONT><BR></TD>
        <TD ALIGN="RIGHT" VALIGN="TOP"><IMG SRC="/iw-icons/
tslogosmall.gif"></TD>
    </TR>
</TABLE>
END

    return;
}

```

```
sub print_footer {  
    print <<"END";  
</BODY>  
</HTML>  
END  
    return;  
}
```



## Appendix B

# Command-Line Tools

---

You can generate or regenerate HTML files from the command line as well as from the TeamSite Templating GUI. Refer to the *TeamSite Templating User's Guide* for information on the GUI.

Both `iwgen` and `iwregen` use an underlying low-level presentation template compiler, called `iwpt_compile.ipl`. This compiler is available for your use and is beneficial when you develop, test, and debug presentation templates.

The presentation template compiler, `iwpt_compile.ipl`, is a command-line tool that uses the data content records, Perl code, and `iw_xml` tags to produce output. You can use the presentation template compiler when you develop new tags.

The `iwtdtd2sym` CLT is used to create data capture templates from industry-standard DTDs. Refer to Appendix C, “Creating DCTs from DTDs” for examples of using this CLT.

The `iwxml_validate.ipl` CLT validates XML files against a DTD.

The `upgrade_dct_cfg.ipl` CLT upgrades `datacapture.cfg` files from regex5 basic regular expression syntax to extended regular expressions.



## iwddctacleval

Alters a data capture template to have only one data capture instance per item, according to ACLs in the data capture template (DCT). It evaluates ACLs (set with <allowed> tags) inside DCTs. It also runs server-side callouts. The templating Java client receives a DCT from the TeamSite server. The document it receives has been through this ACL evaluation process and the server-side inline callout substitutions. This CLT is a debugging tool that lets you see the exact DCT that the client sees, which is not the exact DCT that is in the user's workarea.

### Usage:

```
iwddctacleval [-h|-v] [-c] [-e] -u username -r userrole -w workarea dct
```

### Options:

-h	Displays this usage message.
-v	Displays version number.
-c	Displays the Java class path.
-e	Sends errors to STDOUT.
-u <i>username</i>	Specifies the name of the current data capture end user.
-r <i>userrole</i>	Specifies the role of the current data capture end user.
-w <i>workarea</i>	Specifies a vpath to the current workarea.
<i>dct</i>	Specifies a file-system path to the current data capture template.

### Example:

A data capture template that contains the following section is used:

```
<item name="just chris and andre">  
  <textarea><allowed><cred user="chris" /></allowed></textarea>  
  <text><allowed><cred user="andre" /></allowed></text>  
</item>
```

The CLT:

```
iwddctacleval -u chris -r editor /default/main/WORKAREA/chris /path_to/  
datacapture.cfg
```

issues the following results for this section:

```
<item name="just chris and andre">  
  <textarea><allowed><cred user="chris" /></allowed></textarea>  
</item>
```

However, if you issue the CLT as follows:

```
iwddctacleval -u andre -r editor /default/main/WORKAREA/chris /path_to/  
datacapture.cfg
```

the following results are obtained for this section:

```
<item name="just chris and andre">  
  <text><allowed><cred user="andre" /></allowed></text>  
</item>
```



## iwtdtd2sym

Converts an XML DTD into a skeletal data capture symbol table configuration file. This output must be manually modified before further use. The symbol table configuration file will be written to standard output.

### Usage:

```
iwtdtd2sym [-h|-v] [-c] [-r ruleset-name] [-i itemref-name] dtd-location
```

<code>-h</code>	Displays this usage information.
<code>-v</code>	Displays this command's version number.
<code>-c</code>	Displays the Java class path.
<code>-r <i>ruleset-name</i></code>	Specifies the name of the ruleset in the outputted symbol table configuration file. Default is TeamSite Templating.
<code>-i <i>itemref-name</i></code>	Specifies the name of the itemref in the ruleset in the outputted symbol table configuration file. Default is the name of the first element type declared in the DTD.
<code><i>dtd-location</i></code>	Specifies a system literal, which is a URI referencing an XML DTD. Example URIs are: document.dtd (a file system path) ../path/to/document.dtd (a file system path) http://www.flixml.org/flixml/flixml.dtd (a URL)

### Example:

The following line converts the `simple.dtd` file in and outputs it to `iwtdtd2sym.out`.

```
iwtdtd2sym simple.dtd > iwtdtd2sym.out
```

## iwgen

Generates an HTML file based on a presentation template and a data content record.

### Usage:

```
iwgen [-h|-v] -t templatevpath -r recordvpath vpath
```

### Options:

-h	Displays this usage message.
-v	Displays version number.
-t <i>templatevpath</i>	Specifies a path to a TeamSite Templating presentation template, where <i>templatevpath</i> is either a relative vpath or an archive-rooted vpath. Server-rooted vpaths are not supported.
-r <i>recordvpath</i>	Specifies a path to a TeamSite Templating data content record, where <i>recordvpath</i> is either a relative vpath or an archive-rooted vpath. Server-rooted vpaths are not supported.
<i>vpath</i>	Specifies a path to write the TeamSite Templating generated file, where <i>vpath</i> is either a relative vpath or an archive-rooted vpath. Server-rooted vpaths are not supported.
-e <i>encoding</i>	Specifies the encoding to be used for the generated file.

### Example:

The following example generates an HTML file based on the presentation template `auction.tpl` and the data content record `june_items`. The HTML file is written to the file `june_display.html` in the current workarea. The current working directory is the user's workarea. You should enter this as a single line.

```
% iwgen -t templatedata/internet/auction/presentation/auction.tpl  
templatedata/internet/auction/data/june_items june_display.html
```



## iwprop

Reads or modifies the templating preference in the entity database.

### *Usage:*

To set templating preference:

```
iwprop -user username -s -key teamsite_templating/config/use_java_ui  
-value true|false
```

To get the value of templating preference.

```
iwprop -user username -g -key teamsite_templating/config/use_java_ui
```

The *username* variable is the *userid*. If on a Windows platform, it has to include domain name.

The key with which the templating preference is stored is  
teamsite\_templating/config/use\_java\_ui.

Templating interface preference is role insensitive; that is, a user's templating interface preference is used across the roles. Regardless of the logon role, the same preference applies for that user.

## iwpt\_compile.ipl

Invokes the command-line presentation template compiler to compile presentation templates into output formats such as HTML, jsp, and asp. By default, the output encoding of characters is UTF-8, but it can also be ISO-8859-1, if specified with the `-oenc ISO-8859-1` flag.

By default, the output of this program is the final result of compiling a template. If the `-ofile filename` flag is used, this output is sent to `filename`, otherwise it is printed to STDOUT.

If the `-ocode filename.ipl` flag is used, instead of writing the normal result of the compilation process out to file, a stand-alone program that generates the output is written. This is useful when debugging presentation templates and custom `<iw_xml>`-derived tags).

### Usage:

```
iwpt_compile.ipl -pt filename [-ofile filename] [-ocode filename]  
[-oenc encoding] [-smartwrite] [tag-specific flags]
```

```
iwpt_compile.ipl -v | -h
```

### Arguments:

<code>-v</code>	Prints the version number on STDOUT.
<code>-h</code>	Prints a help message.
<code>-pt <i>filename</i></code>	Use the <i>filename</i> presentation template.
<code>-ofile <i>filename</i></code>	Save the output to <i>filename</i> instead of STDOUT.
<code>-ocode <i>filename.ipl</i></code>	Writes to a stand-alone program named <i>filename.ipl</i> that generates the output.
<code>-oenc <i>encoding</i></code>	Specifies output encoding, which is UTF-8 by default. Specify <code>-oenc</code> on the XML declaration line of the presentation template.
<code>-smartwrite</code>	Specifies <code>-ofile</code> only overwrites <i>filename</i> if it is different.



Tag-specific flags:

<code>-iw_pt-dcr</code>	The file names that follow this <code>iwpt_compile.ipl</code> flag must be a valid data content record. <code>iw_pt</code> reads in the data content record and makes its values available through <code>iw_value</code> .
<code>-iw_pt-arg</code>	The key, value pairs that follow this flag are used to initialize the presentation template arguments within the template. This is useful when debugging a component that normally gets its <code>%iw_arg</code> initialized by the <code>%iw_param</code> of its enclosing template's <code>&lt;iw_include&gt;</code> tag.
<code>-iw_include-location</code>	Mandatory when the mode attribute of the <code>iw_include</code> tag is <code>docroot</code> . The file path is prepended to the file name provided in the <code>file</code> attribute to form a complete file path (used to virtualize the inclusion).

### Example 1

This compilation line uses `iw_pt-dcr` to obtain data from a single data content record named `moo.dcr`.

```
iwpt_compile.ipl -pt -iw_pt moo.tpl -iw_pt-dcr moo.dcr cow.dcr  
-iw_include-location . -ofile moocow.html
```

### Example 2

This example shows output that needs to be in UTF-8 format, and the output file should not be overwritten if the contents have not changed.

```
iwpt_compile.ipl -pt -iw_pt moo.tpl -iw_pt-dcr moo.dcr cow.dcr  
-smartwrite -iw_include-location . -ofile moocow.html -oenc UTF-8
```



### Example 3

This example shows the use of `iwpt_compile` for debugging.

```
iwpt_compile.ipl -pt -iw_pt moo.tpl -iw_pt-dcr moo.dcr cow.dcr  
-iw_include-location . -ocode xxx.ipl ; xxx.ipl
```

The limitations to using `iwpt_compile.ipl` directly are:

- Output pages are not associated with data content records.
- The output pages are editable pages (using SmartContext Editing), but they cannot be accessed through the TeamSite Templating GUI.

When you call the presentation template compiler, you can specify command-line arguments and flags. Command-line flags are specific to and used by various `iw_xml` tags rather than being used directly by the compiler. They are specified as part of the `iwpt_compile.ipl` command.

When a presentation template is processed from the presentation template compiler, the following steps are performed:

1. The presentation template is compiled using the command-line utility `iwpt_compile.ipl`. It may use zero or one XML-based data content records.
2. An XML parser reads the presentation template. As the parser reads, it encounters XML tags.
3. A tag object of the appropriate type is created and the parser calls that object's member functions, passing it relevant information, such as attribute list key, value data.
4. The tag object's member function emits a snippet of Perl.
5. Collectively, all the snippets of Perl that these tag object member functions emit as the parser scans the template form a program.
6. This program runs, and the result is the document (typically HTML) that merges content with look-and-feel instructions.



## iwregen

Regenerates an HTML file that was generated by TeamSite Templating based on a presentation template and a data content record. Use this command to update a generated HTML file if either or both the presentation template and the data content record that the file is based on have been modified.

### *Usage:*

```
iwregen [-h|-v] vpath
```

Options:

<code>-h</code>	Displays this usage message.
<code>-v</code>	Displays version number.
<code><i>vpath</i></code>	Specifies the path to the file that will be regenerated, where <i>vpath</i> is either a relative vpath or an archive-rooted vpath. Server-rooted vpaths are not supported.

### *Example:*

The following example regenerates the HTML file `june_display.html`, which resides in the current workarea.

```
% iwregen june_display.html
```

## iwxml\_validate.ipl

Validates a list of XML files against a DTD (and can also check to see if the XML files are well-formed).

### Usage:

```
iwxml_validate.ipl [-max_errors n] [-d level] [-well] x.xml [y.xml [...]]
```

```
iwxml_validate.ipl -h |-v
```

`-max_errors n`

Displays maximum of *n* errors before quitting XML validation on the current file. The default is to report all errors.

`-d level`

Sets debug verbosity level (where *level* is 0-3); the default debug verbosity level is 2.

#### Debug

##### level

##### Displays

0

Nothing

1

A terse message on failure

2

Parsing warnings and failures

3

Messages on success and failure

`-well`

Checks to see if XML is well-formed, but does not validate.

`-h`

Displays this usage information.

`-v`

Displays this command's version number.

**Example:**

Given an XML file (for example, `x.xml`):

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE a SYSTEM "x.dtd">
<a>
  <b p='c'>this</b>
  <b p='a'>is</b>
  <b p='zzzzzz'>a valid</b>
  <b p='b'>xml file</b>
</a>
```

and a DTD (for example, `x.dtd`):

```
<!ELEMENT a (b*)>
<!ELEMENT b (#PCDATA)>
<!ATTLIST b p CDATA #REQUIRED>
```

the command line:

```
iwxml_validate.ipl x.xml
```

will return with no output and an exit status indicating success since `x.xml` is a valid XML file.

## upgrade\_dct\_cfg.ipl

The `upgrade_dct_cfg.ipl` CLT upgrades `datacapture.cfg` files from regex5 basic regular expression syntax to extended regular expression syntax. The meanings of the original basic regular expressions are preserved, but the extended regex grammar provides more expressive power for validating user input.

This upgrade is required when moving from the browser-based data capture interface of TeamSite Templating Classic 4.5 or from any previous version of TeamSite Templating that uses regular expression syntax. Only validation-regex attributes containing the following characters are affected:  
+ ? | ( )

CAUTION: You should not run this utility more than once on a particular file (see `-force` for details).

### *Usage:*

```
upgrade_dct_cfg.ipl [-log file] [-inplace] [-n] [-d verbosity]
[-no_iwcfg_update] [-force] [-no_staging_update]
[directory_name|file_name]+
```

```
upgrade_dct_cfg.ipl -v | -h
```

`-log file`

The name of the file to which log information is sent. By default, log information is printed on STDOUT. For example, if `-log xxx` is used, all log information is sent to the file named `xxx`.

`-inplace`

Do not make backup copies of the `datacapture.cfg` files; without this switch, `datacapture.cfg.backup` files are placed in the same directories as the `datacapture.cfg` files.

`-n`

Do not write or modify any `datacapture.cfg` files; just determine which ones require an upgrade. Do not modify `/etc/iw.cfg`.



<code>-no_iwcfg_update</code>	Do not modify anything in <code>iw.cfg</code> . By default, running this utility sets <code>use_extended_regex5=true</code> within the <code>[teamsite_templating]</code> section of <code>/etc/iw.cfg</code> .												
<code>-force</code>	This utility should run at most once on the root of TeamSite branching structure (for example, <code>/iwmnt</code> ) since the conversion from basic regexes to extended regexes is one-way. If <code>use_extended_regex5=true</code> is already set within the <code>[teamsite_templating]</code> section of <code>iw.cfg</code> , it is assumed that no further conversion of <code>datacapture.cfg</code> files is required, and this utility will exit with a diagnostic message. To override this behavior, use the <code>-force</code> flag.												
<code>-no_staging_update</code>	Do not attempt to upgrade <code>datacapture.cfg</code> files that are already in the staging area. By default, <code>datacapture.cfg</code> files in the staging area are upgraded by creating a temporary workarea, doing an update of the relevant files, and then automatically checking in the changes.												
<code>-d verbosity</code>	Set the debug verbosity level: <table><thead><tr><th>Verbosity</th><th>Displays</th></tr></thead><tbody><tr><td>0</td><td>Nothing</td></tr><tr><td>1</td><td>Only files requiring upgrade</td></tr><tr><td>2</td><td>Changed and unchanged files (default)</td></tr><tr><td>3</td><td>Information from Level 2 plus low-level trace messages</td></tr><tr><td>4</td><td>Information from Level 3 with extensive trace messages</td></tr></tbody></table>	Verbosity	Displays	0	Nothing	1	Only files requiring upgrade	2	Changed and unchanged files (default)	3	Information from Level 2 plus low-level trace messages	4	Information from Level 3 with extensive trace messages
Verbosity	Displays												
0	Nothing												
1	Only files requiring upgrade												
2	Changed and unchanged files (default)												
3	Information from Level 2 plus low-level trace messages												
4	Information from Level 3 with extensive trace messages												
<code>-h</code>	Displays this usage information.												
<code>-v</code>	Displays this command's version number.												

### **Examples:**

```
upgrade_dct_cfg.ipl $iwmount
```

Runs the utility on all the templating files it finds.

```
upgrade_dct_cfg.ipl -force -no_staging_area  
y:\default\main\www\WORKAREA\work
```

Upgrades the templating files in the named workarea. The `-force` option is specified to override the `use_extended_regex5=true` statement set within `iw.cfg`. The `-no_staging_area` option is specified to save time since staging area files are read-only files.

### **Background:**

In *basic* regular expressions (the old default):

<b>Character</b>	<b>Meaning</b>
------------------	----------------

+	a single instance of the '+' character
?	a single instance of the '?' character
	a single instance of the ' ' character
\( and \)	used for grouping
\{ and \}	used for expressing ranges of instances

In *extended* regular expressions:

<b>Character</b>	<b>Meaning</b>
------------------	----------------

+	one or more instance
?	zero or one instance
	either the left or the right hand alternative
( and )	used for grouping
{ and }	used for expressing ranges of instances

In extended regular expressions, if you wish to use a literal `+`, `|`, `(`, `)`, `{`, or `}` character in your regex, you must escape it with a `\`. For example: The *basic* validation regex `^(hi)\{2,5\}` is written as `^(hi){2,5}` after the conversion to extended regular expressions.



A basic regex like `you+me` must now be expressed as `you\+me` because `+` means *one or more*. Therefore, the extended regex `you+me` matches strings like `youme`, `youume`, `youuume`, etc.

You should probably revisit your validation regexes, since the extended regular expressions now being used allow for stricter input checking.



## Appendix C

# Creating DCTs from DTDs

---

You can create `datacapture.cfg` files that define data capture templates (DCTs) from industry-standard XML DTDs. These data capture templates display as data capture forms in TeamSite Templating. A list of the steps to convert DTDs is outlined here. Refer to the remainder of this appendix for details and examples of the files at each step in the processing of creating the `datacapture.cfg` file.

1. Verify that the DTD is correct.
2. Run the `iwdtd2sym` CLT to convert the DTD.
3. Copy the output from the `iwdtd2sym` CLT to `symbol-table.cfg`.
4. Optionally modify the `symbol-table.cfg` to change the name attribute of the `itemref` sub-element of the `<ruleset>`, and save the file as `datacapture.cfg`.
5. Make any additional edits to add items such as labels and descriptions to `<items>` and save the file.
6. Identify the new `datacapture.cfg` file in the `templating.cfg` file.

Save all your intermediate output files along with the DTD and the final `datacapture.cfg` file. It is recommended that these files be versioned in TeamSite.



## Running the CLT on the DTD File

The following file is a sample DTD, named `simple.dtd`.

```
<!-- This is a simple example DTD.  
      It is a "Hello, world!" type of DTD.  
-->  
  
<!ELEMENT simple-example (message)>  
      <!ATTLIST simple-example  
        color      (red|blue|green)      #IMPLIED  
      >  
  
<!ELEMENT message      (#PCDATA)>
```

Run the `iwtdtd2sym` CLT on the DTD, specifying the complete path to the DTD, to create the file that begins on the next page by changing to the directory containing the DTD:

```
cd Y:\default\main\WORKAREA\chris\templatedata\internet\simple-example
```

(the reference to the Y: drive is not needed for UNIX platforms) and issuing the command:

```
iwtdtd2sym simple.dtd > iwtdtd2sym.out
```

Refer to the Appendix B, “Command-Line Tools” for additional details on `iwtdtd2sym`.

## The datacapture.cfg File

The following file is the output from the iwddtd2sym CLT (iwddtd2sym.out), which has been copied to a file named datacapture.cfg and then edited.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data-capture-requirements SYSTEM "datacapture5.0.dtd">

<data-capture-requirements dtd-system-identifier="simple.dtd" name=""
type="content">
  <symbol-table>
    <symbol name="simple-example">
      <container combination="and" hide-name="f" name="simple-example">
        <itemref name="{iw_attributes}"/>
        <container combination="and" hide-name="t"
name="{iw_sub_elements}[0]">
          <label>XML sub-elements</label>
          <itemref name="message"/>
        </container>
      </container>
    </symbol>
  <symbol regex="^(.*)?simple-example/{iw_attributes}$">
    <container combination="and" hide-name="t"
name="{iw_attributes}">
      <item name="color">
        <select multiple="f" required="f" size="0" width="0">
          <option label="red" selected="f" value="red"/>
          <option label="blue" selected="f" value="blue"/>
          <option label="green" selected="f" value="green"/>
        </select>
      </item>
    </container>
  </symbol>
```

Reference to simple.dtd maintained.<sup>1</sup>

XML elements entered as symbols.<sup>2</sup>

A <container> contains an <itemref>.<sup>3</sup>

A set of attributes also become a <symbol>.<sup>4</sup>

The color attribute.<sup>5</sup>



```
<symbol name="message">
  <container combination="and" hide-name="f" name="message">
    <container combination="and" hide-name="t"
      name="{iw_sub_elements}[0]">
      <label>XML sub-elements</label>
      <item name="#PCDATA">
        <textarea cols="0" required="f" rows="0" rtf="f"
          wrap="off"/>
      </item>
    </container>
  </container>
</symbol>
</symbol-table>

<ruleset name="This is my only rule">
  <itemref name="simple-example"/>
</ruleset>

</data-capture-requirements>
```

The message element as a #PCDATA item.<sup>6</sup>

Editing the ruleset name.<sup>7</sup>

## Diagram Key

1. This file maintained a reference to the DTD from which it originated, in the `dtd-system-identifier` attribute of the `data-capture-requirements` element.
2. This file was generated directly from an industry-standard XML DTD. Each XML element becomes a `<symbol>` element in the `<symbol-table>`.
3. Every element type declared in the DTD is represented in its `<symbol>` as a `<container>`. A `<container>` that represents an XML element type will contain an `<itemref>` element for the element type's attributes, if any. A `<container>` that represents an XML element type will contain another `<container>` for its subelements. A `<container>` that represents a set of the subelements of an XML element type will contain an `<itemref>` reference for each subelement type it refers to. This XML element type (`simple-example`) has a simple content specification (`message`), so there is just one `<itemref>`.
4. The set of attributes for each element type also becomes a `<symbol>`.
5. The set of attributes of an element type is represented in its `<symbol>` as a `<container>`. There is only one attribute, `color`. Because it was an enumerated attribute, it is represented here by a `<select>` element.
6. Here is the `message` element type. Its content specification was also simple: `#PCDATA`. A character data reference in the DTD is transformed into a data capture `<item>` named `#PCDATA`.
7. The name attribute of the `itemref` subelement of the `<ruleset>` defaults to the name of the first element type declared in the DTD. The ruleset contains a single `<itemref>`. The `itemref` name defaults to the `symbol name`. This `<itemref>` references the outermost element of the XML documents that will be generated as DCRs. In this example, the `ruleset name` was edited.

You may manually add items such as labels and descriptions to this file. For examples, you may want to add `<label>` and `<description>` elements to `<items>` and `<containers>`, and to specify `<min>` and `<max>` values in a `<replicant>` element. You can also modify instances.



## Unsupported DTD Features

A few features in DTDs are not supported by the CLTs and the conversion process:

- An element `<section>` that can legally, according to the DTD, contain another `<section>` element is not supported to an arbitrary depth. The data capture template author must decide the depth to which an element can recursively contain elements of the same type. This is done with a regex on the `<symbol>` element.

An example of this section of a DTD is:

```
<!ELEMENT   body    (section)*>
<!ELEMENT   section (title|paragraph|sub-section)*>
<!ELEMENT   subsection (section)*>

...
```

The following is an example of a regex that captures a `<section>` element with another `<section>` element:

```
<symbol regex="^(.*)?section/(.*)?section$">
```

The following is an example of a regex that captures a `<section>` element with another `<section>` element within another `<section>` element:

```
<symbol regex="^(.*)?section/(.*)?section/(.*)?section$">
```

The first regex would catch a two-deep nesting level (anything greater than or equal to two), and the second regex would catch any nesting level 3 or greater. The regex `<symbol name="section">` would catch all levels.

Therefore, these symbols need to be ordered in the `symbol-table` by depth; for example:

```
<symbol regex="^(.*)?section/(.*)?section/(.*)?section$">
...
</symbol>
<symbol regex="^(.*)?section/(.*)?section$">
...
</symbol>
<symbol name="section">
...
</symbol>
```

- The validity constraints for the ID, IDREF, IDREFS, ENTITY, ENTITIES, NMTOKEN, and NMTOKENS attribute types are not enforced.

For an explanation of the validity constraints, refer to section 3.3.1 of the XML 1.0 specification, located at <http://www.w3.org/TR/1998/REC-xml-19980210>.





## Appendix D

# Internationalization

---

Data capture templates can include multi-byte text (in field names and field descriptions). Also, this multi-byte text can be in an encoding other than UTF-8. UTF-8 encoded data capture templates that include multi-byte text are processed correctly.

The character encoding of data capture templates can be in any encoding that the Xerces XML parser can parse. These are:

- UTF-8 (Multi-Octet Unicode)
- ISO-8859-1 (Western European)
- ISO-8859-2
- ISO-8859-3
- ISO-8859-4
- ISO-8859-5
- ISO-8859-6
- ISO-8859-7
- ISO-8859-8
- ISO-8859-9
- gb2312 (Simplified Chinese)
- EUC-JP (Japanese - Extended Unix Code)
- iso-2022-jp (Japanese - 7 bit encoding)
- Shift-JIS (Japanese - Most common in Japan)
- Big5 (Traditional Chinese)
- euc-kr (Korean - Extended Unix Code)
- koi8-r (Russian)
- ebcdic family of encodings



Note: Interwoven has certified data capture templates encoded in ISO-8859-1, UTF-8, Shift-JIS, and EUC-JP. The other encodings should work, but they have not been certified.

These encoding names need to be used literally in XML files (`datacapture.cfg`). For example, to include Shift-JIS text in the `datacapture.cfg` file, the xml header should be:

```
<?xml version="1.0" encoding="Shift-JIS" standalone="no"?>
```

The following are incorrect specifications: `encoding="SJIS"`, `encoding="shift-jis"`, or `encoding="CP932"`. Use `encoding="Shift_JIS"`.

## Limitations

Data capture templates can be encoded in non-UTF-8 encoding if these data capture templates are not used by DataDeploy in DAS mode. DataDeploy in DAS mode parses data capture templates to generate column names in `dd.cfg` (DataDeploy configuration file). DataDeploy's XML parser is limited to parsing two encoding sets: UTF-8 and ISO-8859-1 encoded data capture templates. If data capture templates need to include non-ASCII multi-byte field names and need to be used by both DataDeploy and TeamSite Templating, use UTF-8 encoded data capture templates. For DataDeploy in non-DAS mode, ensure that your `dd.cfg` is encoded in UTF-8 if it contains multi-byte column names.

When entering multi-byte characters (for example, Japanese, Chinese, or Korean) in TeamSite Templating Java, the client operating system must be operating in the proper locale for correct character input. For example, the following situations would be acceptable for entering Japanese into data capture templates:

- Client is Japanese Windows 98, ME, NT or 2000 using the Microsoft IME (MSIME).
- Client is Windows 2000 English version with its default locale set to Japanese (refer to your Windows manuals for instructions on how to do this).

## Japanese EUC-JP Encoding Support

EUC-JP encoding is an additional supported encoding for presentation templates, and the Templating output engine. Templating can now generate HTML in EUC-JP encoding. Presentation templates can also be in EUC-JP encoding. Also, templating tags such as `<iw_include>` that have supported encoding in previous TeamSite Templating versions now support EUC-JP encoding.

The list of encodings certified for Templating output engine (presentation templates and page generation engine, and `iw_include` tags) are:

- CP932          Superset of Shift-JIS. Japanese PC encoding. (CP932 is also known as MS-SJIS.)
- CP936          Superset of GBK. GBK is a superset of GB2312, which is a superset of GB2312-80. (GB2312 is Simplified Chinese encoding, used in China, Singapore.)
- CP950          Superset of Big5. Traditional Chinese encoding pervasive in Taiwan, Hong Kong.
- CP949          Encodes KSX-1001-1997, a superset of KSC-5601-1992 - Korean National Standard. (Korean PC encoding.)
- ISO-8859-1    Western European encoding. Also known as Latin 1.
- UTF-8          Multi-octet Unicode. Encodes the Unicode character set.
- EUC-JP          Extended Unix Code - Japan. Japanese encoding common on UNIX platforms such as Solaris 2.x.

To include multi-byte text in presentation templates in one of the above encodings, the encoding name needs to be specified as listed. For example, if a particular presentation template is encoded in Shift-JIS encoding, the xml header for this presentation template needs to be:

```
<?xml version="1.0" encoding="CP932" standalone="no"?>
```

To customize for page generation of HTML output in particular encodings, refer to [http://iw-home/iw/help/tst/pt/iwpt\\_encoding.html](http://iw-home/iw/help/tst/pt/iwpt_encoding.html) and [http://iw-home/iw/help/tst/pt/iwpt\\_compile.html](http://iw-home/iw/help/tst/pt/iwpt_compile.html).

There is no restricted binding between encoding of a data capture template, encoding of a presentation template, and encoding of page generation. Data capture templates can be in one encoding, while presentation templates be in a different encoding, and HTML output in yet another encoding. For example, you could have a data capture template created/edited in Shift\_JIS. The data



content records can then be combined with an UTF-8 encoded presentation template, to generate an EUC-JP encoded HTML file. The only constraint is that data content records are encoded in UTF-8.

The presentation template embeds HTML with a `<META HTTP-EQUIV>` header such as the following:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8">
```

The `charset` parameter needs to be edited to correspond with the chosen encoding of page generation. For example, if the default encoding in `iwpt_compile.ipl` has been changed to generate pages in Shift-JIS encoding, then the `META HTTP-EQUIV` header inside the presentation template needs to specify:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Shift-JIS">
```

This `charset` specification is not necessarily the same as the `encoding` specification in the `xml` header of the presentation template itself. For example, if the presentation template itself needs to be encoded in EUC-JP, but page generation needs to produce Shift-JIS encoded pages, then the `xml` header of the presentation template would specify:

```
<?xml version="1.0" encoding="EUC-JP" standalone="no"?>
```

but the `META HTTP-EQUIV` header of HTML inside the presentation template would specify:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Shift-JIS">
```

In this scenario, any Japanese text in the presentation template needs to be in EUC-JP encoding, even if the text is part of the HTML segments. When page generation takes place (by `iwpt_compile.ipl`), all text is then converted to the chosen encoding as specified in `iwpt_encoding.ipl` (refer to [http://iw-home/iw/help/tst/pt/iwpt\\_encoding.html](http://iw-home/iw/help/tst/pt/iwpt_encoding.html) and [http://iw-home/iw/help/tst/pt/iwpt\\_compile.html](http://iw-home/iw/help/tst/pt/iwpt_compile.html) for more information).

## **Localized Java Templating into Japanese, Traditional Chinese, and Simplified Chinese**

A Localized Java Templating interface is available in three languages: Japanese, Traditional Chinese, and Simplified Chinese. Depending on the localized system in which Templating runs, the corresponding localized Templating interface is presented. For example, the TeamSite server with TeamSite Templating runs on an U.S-English system. A Japanese Templating user logs on to the TeamSite server from a Japanese Windows NT system. When Templating is invoked, all Templating user interfaces would be presented in Japanese. (However, the TeamSite user interface is still in English.) Similarly, if a Templating user logs in from a Traditional Chinese OS, the user would get a Traditional Chinese Templating Client user interface.



# Index

---

## A

- ACLs
  - evaluating 130
- adding replicants 56
- allowed element 52, 60, 79, 102, 106
- and element 52, 80, 102, 106
- architecture
  - TeamSite Templating 18
- author\_submit\_dcr.wft 109
- available\_templates.cfg 108
  - editing 33
- available\_templates.ipl 32

## B

- branch element 103, 105
- browser element 51, 78

## C

- callout button 116
- callout CGI program 119
- callout element 52, 80
- callouts
  - CGI 116
  - datacapture.cfg example 112
  - Java 111
- category 23
- category element 101, 105

## CGI callout

- data capture form 116
- datacapture.cfg 118
- datacapture\_callout.ipl 119
- cgi-callout element 53, 80
- checkbox element 54, 78

## CLT

- iwdctacleval 130
- iwdtd2sym 132, 146
- iwgen 133
- iwprop 134
- iwpt\_compile.ipl 135
- iwregen 138
- iwxml\_validate.ipl 139
- upgrade\_dct.cfg.ipl 141
- component directory 25
- component template 85
  - example 87
- configuration files
  - available\_templates.cfg 33
  - available\_templates.ipl 32
  - datacapture.cfg 22, 24
    - example 45, 72
  - DataDeploy 109
- overview 22
- presentation template 25
- templating.cfg 22, 27, 32, 97
  - example 98

## content

- creating 27
- creating records 20
- cred element 52, 79, 102, 106

## D

- data capture callout CGIs 120
- data capture form 42
  - example 70
  - with callout button 116
- data capture subsystem 20, 21
- data capture symbol table
  - creating 132
- data capture template
  - creating from DTDs 145
  - customizing 41
  - definition 20
  - example 42
  - overview 40
- data capture templates
  - encoding 153
- data category 23
  - making available 101
- data content record
  - creating 27
  - definition 21, 24
  - example 68, 70, 87
  - initiating workflow 33
  - searching 35

- data directory 24
- data type 23
  - making available 101
- database element 49, 81
- datacapture.cfg 22, 24, 27, 40, 45, 145
  - callout CGI capability 118
  - example 72
- datacapture\_callout.ipl 119
- data-capture-requirements
  - element 47
- DataDeploy
  - configuration files 109
  - integrated with TeamSite Templating 108
  - running as a workflow job 107
- data-type element 101, 105
- debugging tags 135
- default element 80
- deleting replicants 56
- description element 48
- directory element 103, 106
- directory structure
  - contents 24
  - copying 32
  - overview 23
  - sample 30
- DTD
  - converting to data capture templates 145
  - data capture 76
  - sample 146
  - unsupported features 150

**E**

- element
  - allowed 52, 60, 79, 102, 106
  - and 52, 80, 102, 106

- branch 103, 105
- browser 51, 78
- callout 52, 80
- category 101, 105
- cgi-callout 53, 80
- checkbox 54, 78
- cred 52, 79, 102, 106
- database 81
- data-capture-requirements 47
- data-type 101, 105
- default 80
- description 48
- directory 103, 106
- inline 54, 64, 82
- item 48, 77
- java-callout 53, 80, 111
- locations 105
- not 52, 80, 102, 106
- option 54, 79
- or 52, 80, 102, 106
- presentation 102, 105
- radio 55, 79
- replicant 56, 79
- ruleset 48, 77
- select 57, 79
- template 102, 105
- templating 101, 105
- text 58, 77
- textarea 59, 78
- encoding 153
  - Japanese 155
- evaluating ACLs 130
- example templating
  - environment 30, 31
  - copying 32

**F**

- files 45
  - available\_templates.ipl 32
  - datacapture.cfg 22, 40, 72, 145
  - datacapture\_callout.ipl 119
  - DTD 76, 146
  - sample output from iwtdtd2sym CLT 147
  - templating.cfg 22

**G**

- generated HTML files 20, 28
- specifying locations 103

**H**

- hardware requirements 11
- HTML pages
  - from presentation template compiler 135
  - generating 28, 133
  - regenerating 138

**I**

- inline element 54, 64, 82
- installation
  - on Solaris 12
  - on Windows NT 14
- instance
  - defined 41
- integrating 33
- interface
  - controlling 134
- item element 48, 77
  - defined 41



- iw.cfg
  - adding custom menu items 35
  - changing the templating directory 34
  - enabling Java templating 34
  - specifying search paths 35
- iw\_xml tags 96
- iwdctacleval 130
- iwdtd2sym 132, 146
- iwgen 133
- iwprop 134
- iwpt\_compile.ipl 135
- iwregen 138
- iwxml\_validate 139

## J

- Japanese encoding 155
- Java callout 111
  - datacapture.cfg 112
  - source code 112
- java-callout element 53, 80, 111

## L

- locations element 105

## N

- not element 52, 80, 102, 106

## O

- option element 54, 79
- or element 52, 80, 102, 106

## P

- page generation subsystem 20, 21, 29
- presentation directory 25
- presentation element 102, 105

- presentation template 25
  - compiler 22, 85, 135
  - definition 21
  - example 87
  - guidelines 87
  - mapping 102
- previewing data 103

## R

- radio element 55, 79
- replicant element 56, 79
- replicants
  - adding 56
- ruleset element 48, 77
  - defined 40

## S

- search menu item 35
- search paths 35
- select element 57, 79
- software requirements 11

## T

- tags
  - debugging 135
- TeamSite Templating model 18
- template
  - component 85
- template element 102, 105
- templatedata directory 24
  - copying to workarea 32
- templating directory
  - changing 34
- templating element 101, 105
- templating environment
  - example 31

- templating.cfg 22, 27, 32
  - customizing 97
  - DTD 105
  - example 98
- text element 58, 77
- textarea element 59, 78
- type 23

## U

- upgrade\_dct\_cfg 141
- user interface
  - setting 34
- user preference 134

## V

- validating XML 139
- validation regexes 49, 58
  - upgrading 12, 141

## W

- workflow 33
  - DataDeploy process 108
  - initiating 108
  - integrating with TeamSite Templating 108
  - preconfigured 109
  - schematic of 26, 28

## X

- XML
  - validating 139

